

AD-A092 224

WHITE SANDS MISSILE RANGE NM INSTRUMENTATION DIRECTORATE
AUTOMATED READING OF VIDEOTAPE. (U)

F/G 9/2

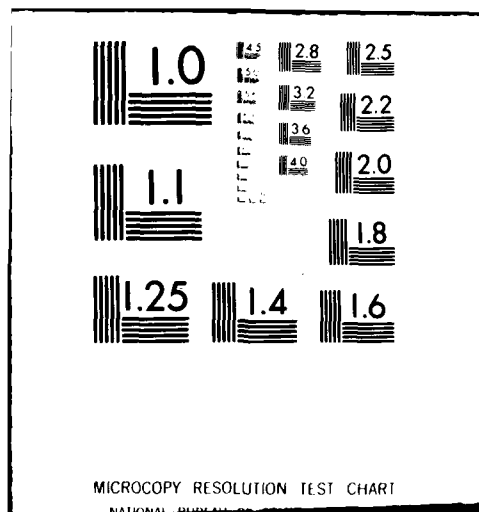
UNCLASSIFIED

NOV 80 R 6 MACHUCA
STEW5-ID-80-2

NL

1 of 1
AD-A092 224

END
DATE
FILMED
1-81
DTIC



AD A092224

LEVEL

12

TECHNICAL REPORT

STEWS - ID - 80 - 2

AUTOMATED READING OF VIDEOTAPE

NOVEMBER 1980

FINAL REPORT

DTIC
SELECTED
DEC 2 1980
C

Approved for public release ; distribution unlimited

INSTRUMENTATION DIRECTORATE
US ARMY WHITE SANDS MISSILE RANGE
WHITE SANDS MISSILE RANGE, NEW MEXICO 88002
8011 24 058

FILE COPY

Destroy this report when no longer needed. Do not return it to the originator.

DISCLAIMER

The findings of this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER STWS-ID-80-2	2. GOVT ACCESSION NO. AD A092224	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AUTOMATED READING OF VIDEOTAPE	5. TYPE OF REPORT & PERIOD COVERED Final Report	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Raul G. Machuca - STWS-ID-T, WSMR, NM	8. CONTRACT OR GRANT NUMBER(s)	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Commander US Army White Sands Missile Range ATTN: STWS-ID-T White Sands Missile Range, New Mexico 88002	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS DA OM 1511 DA Project No. 1L161101A91A	
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE November 1980	13. NUMBER OF PAGES 69
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Image processing Fourier Descriptors Pattern Recognition Video Tracking		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) -This report describes the concepts and software which are used in a proto- type videotape reaper being developed at WSMR. The goal is to read from one frame of video the position of the target and the three angular rotations of the target. This is done by first making a contour of the target from a frame of video. The Fourier descriptors of the contour are then computed and normalized.		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED 184230
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ACKNOWLEDGEMENTS

The author would like to acknowledge those who helped with this report. Ms. P. Smith assisted with the programming. Mr. J. Wise allowed us the use of the laboratory, and Mr. C. Klaassen, as Systems Manager, was very helpful in making the equipment available to us. Ms. H. Essary prepared the final report, with help from Mr. M. Ramos in preparing the figures.

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

TABLE OF CONTENTS

	<u>Page</u>
INTRODUCTION -----	1
SYSTEM DESCRIPTION -----	1
STRUCTURE OF THE PREPROCESSING SOFTWARE -----	9
CONSTRUCTION OF A CONTOUR FROM THE MOMENT FILE -----	16
FOURIER DESCRIPTOR METHODS -----	42
Appendix A. FINDING EDGES IN NOISY SCENES -----	A-1
Section 1. Edges from Moments -----	A-1
Section 2. Second Order Edges -----	A-5
Section 3. Algorithms for Implementation -----	A-6
a. Calculation of Moment -----	A-6
b. Calculation of the Rotation -----	A-8
Section 4. Evaluation -----	A-9

LIST OF ILLUSTRATIONS

	<u>Page</u>
Figures 1a. Digitized video images of F102 -----	3
1b. Result of processing original with an edge detector -----	3
1c. Contour of F102 -----	3
2. Line segments and associated segment numbers ---	4
3. Library of contours of F102. This library was generated from a computer made three- dimensional model -----	5
4. Some contours found by the computer before being corrected by the operator -----	6
5. Contours of Figure 4 after being processed by the operator -----	7
6. The best match found by the computer -----	8
7. Structure of the preprocessing software -----	10
8. Generic hardware model for preprocessing hardware -----	15
9a. F102 flying by a mountain -----	17
9b. F102 flying in front of a mountain -----	17
9c. Hawk missile -----	17
9d. F102, sunspots on wings and nose -----	17
10a. Moment file of F102 flying by a mountain -----	18
10b. Moment file of F102 flying in front of a mountain -----	18
10c. Moment file of Hawk missile -----	18
10d. Moment file of F102, sunspots on wings and nose -----	18
11. Raw histogram Data (I) from moment file. Integrated Data (II) from raw histograms ----	19
12a. Contour found from Moment file (Fig. 10a) -----	20
12b. Contour found from Moment file (Fig. 10b) -----	20
12c. Contour found from Moment file (Fig. 10c) -----	20
12d. Contour found from Moment file (Fig. 10d) -----	20
13. Two-dimensional number patterns and their assigned geometrical directions -----	24

LIST OF ILLUSTRATIONS (cont)

Page

Figure 14. Examples of geometric relations for various predicates, the geometric situation when (a) $P = \text{true}$, and (b) $Q = \text{true}$ -----	27
15. Data structures (a), (b) and operations with their corresponding (c), (d) geometric structures and geometric operations -----	34
16. Geometric criteria for the continuation of a polygonal segment -----	36
17. Polygon before closing by distance measure ----	37
18. Polygon after closing by distance measure ----	38
19. Polygons and corrections entered by the operator -----	43
20. New polygons obtained from corrected polygons of Figure 19 -----	44
21. Three different contours of an F102 from video taken at WSMR -----	45
22. Three basic geometrical shapes and their Fourier coefficients -----	46
23. Contours generated by functions of the type $Z(t) = \exp(it) + i/L \exp(iLt)$ -----	49
A-1. Example center of mass vectors -----	A-2
A-2. Rocket and results of processing -----	A-3
A-3. A curve Γ and its corresponding vector field $\phi(t)$ -----	A-4
A-4. Vector Field at a step or ramp edge point ----	A-5
A-5. Vector field at a roof edge point -----	A-6
A-6. Original of roof edge and edge points -----	A-7
A-7. Clear edges and edges with noise added -----	A-10
A-8. Roof edges with corresponding ROC curves -----	A-11
A-9. Comparison of Sobel and Moment operator -----	A-12
A-10. Signal to noise ratio vs. index of detectability -----	A-13

INTRODUCTION

This report contains a description of an experimental videotape reading system developed at the White Sands Missile Range Instrumentation Directorate computer lab for the investigation of image processing and pattern recognition concepts. The VRS is presently being used to study the concept of accurately determining the aspect angles of a target from one frame of video. The ability of accurately finding the position of a target from one frame of video is useful in extracting a data product from a videotape when there is tape available from only one station. Such a system, made into a real-time hardware machine, would also have applications in fire control of high-energy lasers, since the aiming of such devices requires that exact knowledge about the position of the target be available so that energy can be deposited at a critical point of the target. This experimental system is useful as a test bed for concepts that will have applications both in extracting a data product to be used by customers of WSMR and as a model for a hardware machine that would be used both for real-time tracking and for fire control of new weapons technology.

SYSTEM DESCRIPTION

The data flow of this system is as follows:

- A video tape of a mission is taken at a station.
- The frames to be read are put on a video disk which is attached to an image analyzing system capable of digitizing the frames in the video disc.
- These frames are digitized and put into data files with another file containing all the file names of the frames which are of interest.

The software then processes the data in the following sequence:

1. Read in file containing names of files to be processed.
2. Read in first file and do preprocessing on it until completely done.

begin
repeat

cobegin
begin

- 3a. Read next file and do preprocessing on it.

end;
begin

- 3b. Make a contour of the previously processed file and do the classification.

end;
end;
until eof;

4. Finish off classifying last one read in

end.

Processing begins by first doing pixel level operations. The classification is done by making a line drawing of the plane or rocket to be analyzed, and comparing it against a previously stored line drawing library made from views of the object in question at different angles. Before a contour of the target (Fig. 1a) can be made, points which are possible candidates for edge points must be identified. Since, typically, scenes that we process are very noisy, we begin by doing a three-by-three averaging to every point in the scene. After this, a moment edge detector is used to assign to each point in a scene a value which reflects the probability that a point is an edge point (Fig. 1b). A threshold is chosen by the operator and all points classified as possible target points are assigned a zero and all others a one. The computer then makes and displays a contour of the entire scene with different polygonal segments being assigned different values (Fig. 2). The operator chooses the number of segments which make up the target, and the computer writes the segments out in a file. This file is then modified by the use of interactive graphics programs (Figs. 4 and 5). The result (Fig. 1c) is compared against the library of stored views (Fig. 3), the best match is found (Fig. 6) and the angular data needed is read from the coordinate system. A description of the operations that take place is thus:

begin

1. Read in file containing names of files to be processed.
2. Read in first file and do preprocessing on it until completely done.

repeat

cobegin

begin

- 3A. Read next file and do preprocessing on it.

end;

begin

- 3B. Make a contour by the following process:

- a. Using a histogram, computer chooses a threshold for the moment file of the original and displays a contour based on this threshold.

- b. Is this contour acceptable?

- c. while contour not acceptable do

begin

*Obtain new threshold from operator.

*Draw contour

*Is contour acceptable?

end

- d. Let operator choose segments that will be used to construct target.

- e. Display segments chosen by the operator and modify them as the operator instructs.

- f. Calculate the Fourier descriptors, normalize and do classification.

end;

until eof;

4. Finish off classifying last one read in

end.



Figure 1a. Digitized video image of F102 .



Figure 1b. Result of processing original with an edge detector.

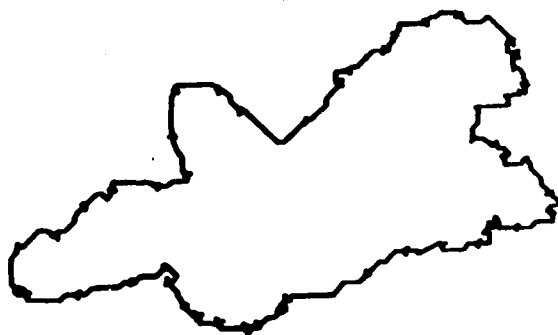


Figure 1c. Contour of F102.

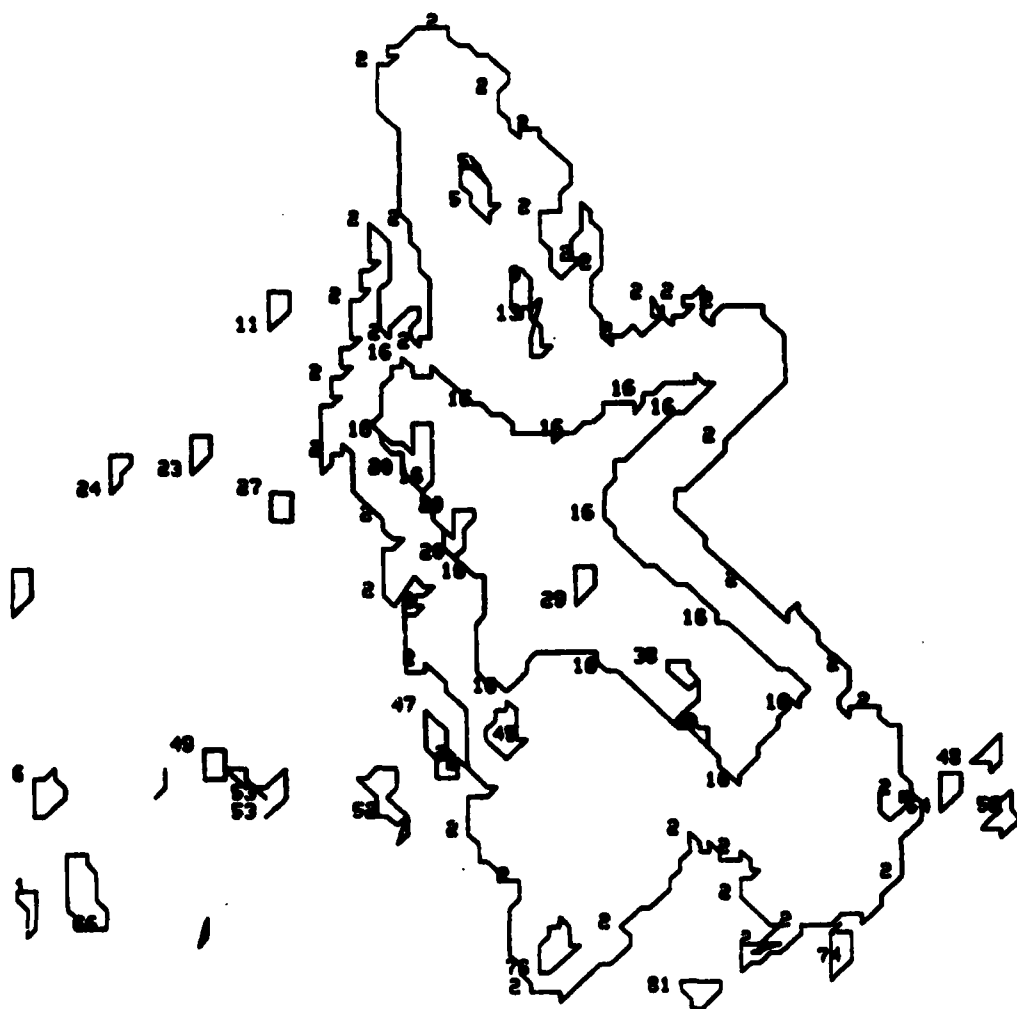


Figure 2. Line segments and associated segment numbers,

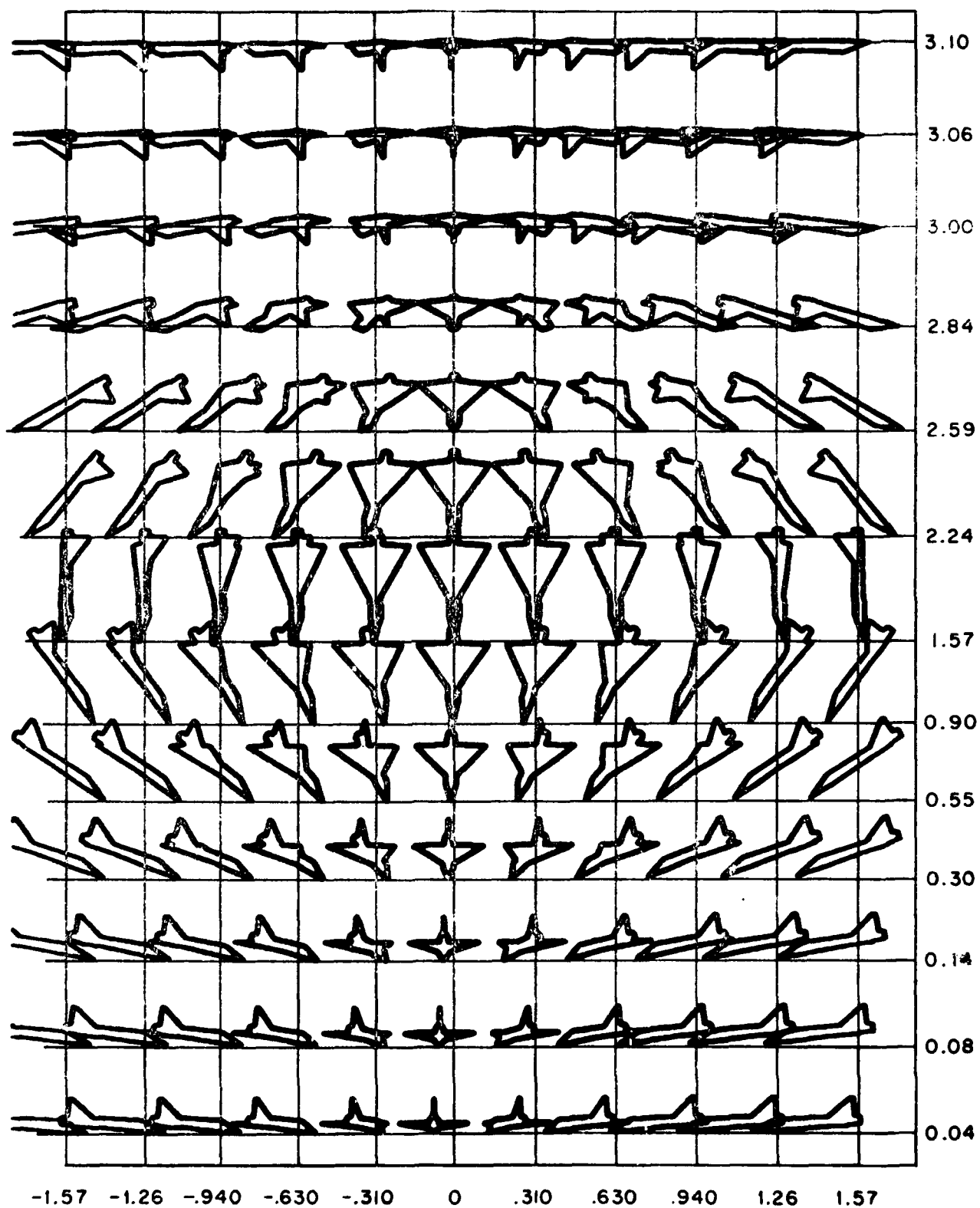


Figure 3. Library of contours of F102. This library was generated from a computer made three-dimensional model.

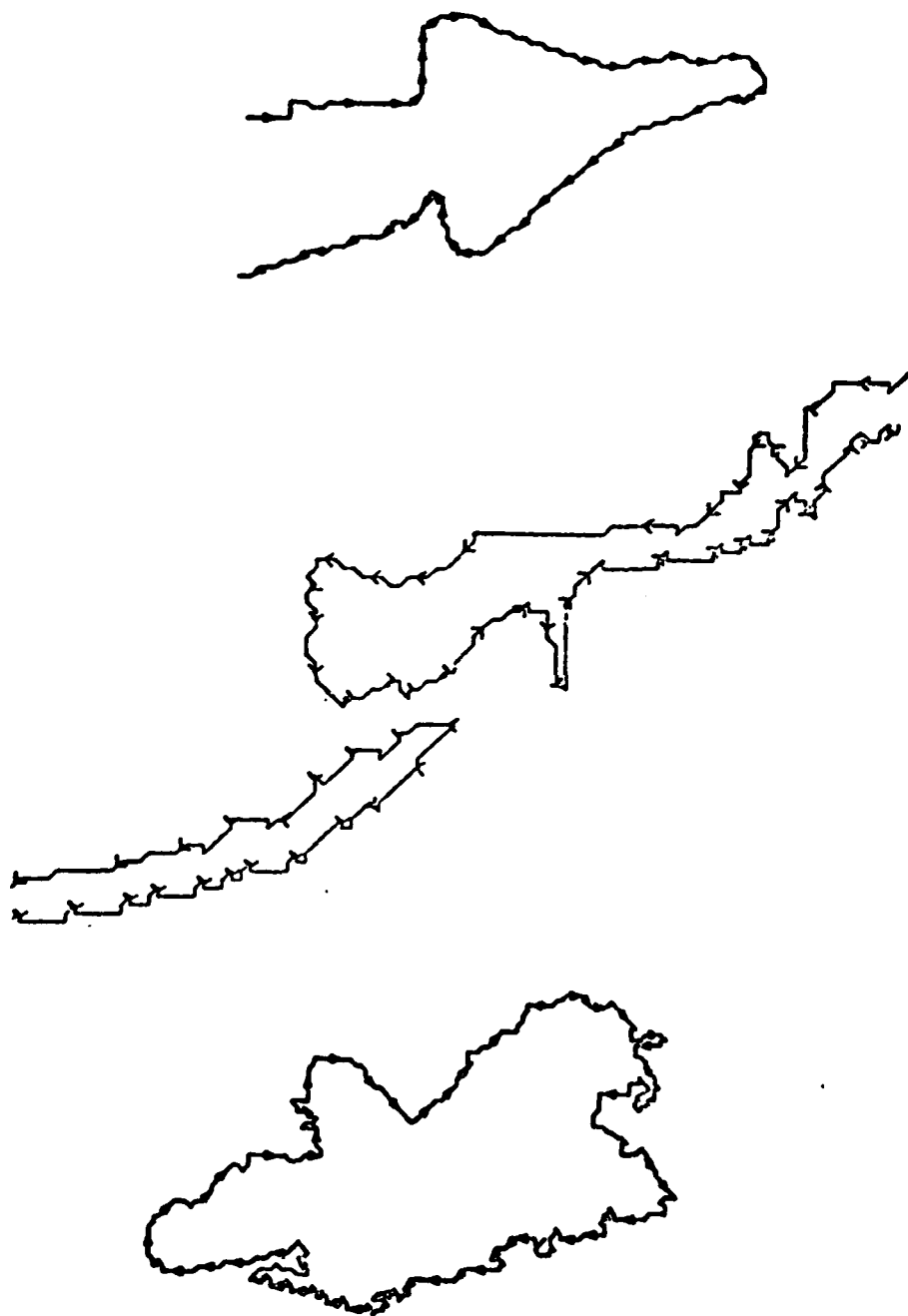


Figure 4. Some contours found by the computer before being corrected by the operator.

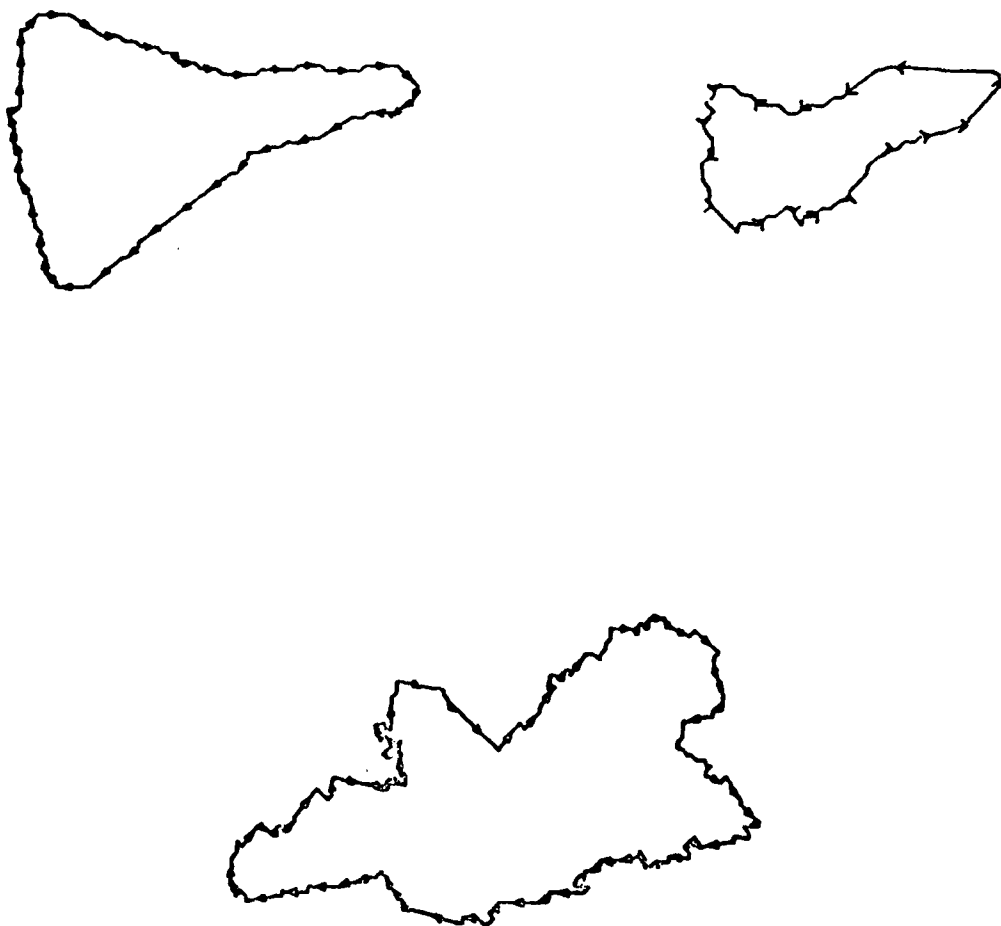


Figure 5. Contours of Figure 4 after being processed by the operator.

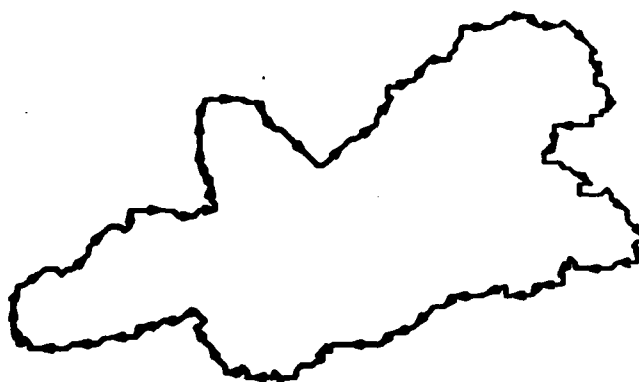
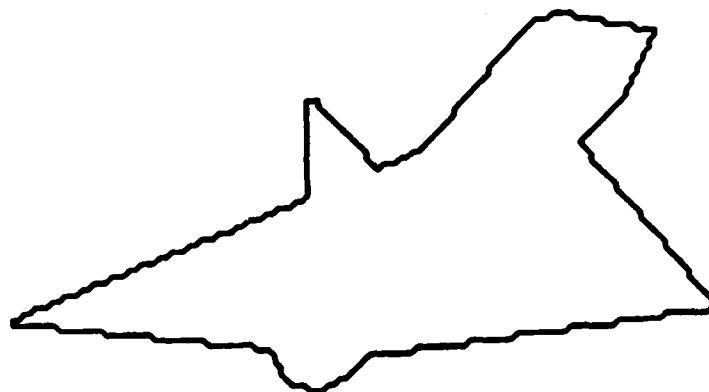


Figure 6. The best match found by the computer.

STRUCTURE OF THE PREPROCESSING SOFTWARE

Before a contour of the scene can be made, points which are possible candidates for edge points must be identified. Since, typically, the scenes that we process are very noisy we begin by doing a three-by-three, averaging to every point in the scene. After this a moment edge detector is used to assign to each point in a scene, a value which reflects the probability that a point is an edge point. As shown in Appendix A this sequence of steps increases the probability of detecting edge points. The next step is to do another averaging operation on the moment file with the purpose of increasing the connectedness of the edge points.

The software to accomplish the preprocessing was written with two ends in mind; one was that this software would be a model for a hardware module to be built later, and the other was that execution time be reduced by overlapping input/output with processing. Figure 7 illustrates how the software is set up. The programs READ 1, WRITE 1 and PROCESS are passive programs in that they suspend themselves immediately after doing some initialization operations. These consist of bookkeeping operations such as setting input file name, output file name, and setting up parameters so that the proper buffer is accessed each time a program is activated. The program which drives these passive programs is called MAIN 1. It runs the needed programs and synchronizes them via the use of global event flags. After the preprocessing is finished it initiates the next step in processing by its call to ARROWS.

A typical frame is processed by MAIN 1 in the following way: First the programs READ1, WRITE1 and PROCESS are loaded into memory. They do whatever initialization is necessary and then suspend themselves. There are two input buffers that will be used by READ1 to store the data to be processed, and two output buffers where processed data is put and from where the program WRITE1 writes the data out onto the disk. MAIN1 first has the two input buffers (lines 15 - 18) filled by the two activations of READ1 done by two calls to RESUME(READ1). READ1 automatically processes the buffers in an alternate manner as do PROCES and WRITE1. The buffers are initially set up (lines 21 - 24) so that the remaining processing can be done concurrently (lines 25 - 35). In the do loop there are waits for flags to be set that indicate that each of the programs involved are finished. The rest of MAIN1 finishes up with the buffers that need to be processed and written out. On line 45 it starts the next step for this frame by its call to ARROWS.

```
0015          CALL RESUME(READ1)  !  FILL IN BUFFER #2
0016          CALL WAITFR(36)
0017          CALL CLREF(36)
0018          CALL RESUME(READ1)
0019          CALL WAITFR(36)
0020          CALL CLREF(36)  !  BUFFERS #1 and #2 FULL
          C
0021          CALL RESUME(PROCE1,)
0022          CALL WAITFR(37)
0023          CALL CLREF(37)
          C
          C  AT THIS POINT INBUF#1 AND INBUF#2 ARE FILLED
          C  AND #2 HAS BEEN COPIED OVER TO OUTBUF #2
          C
0024          CALL WAITFR(42)
```

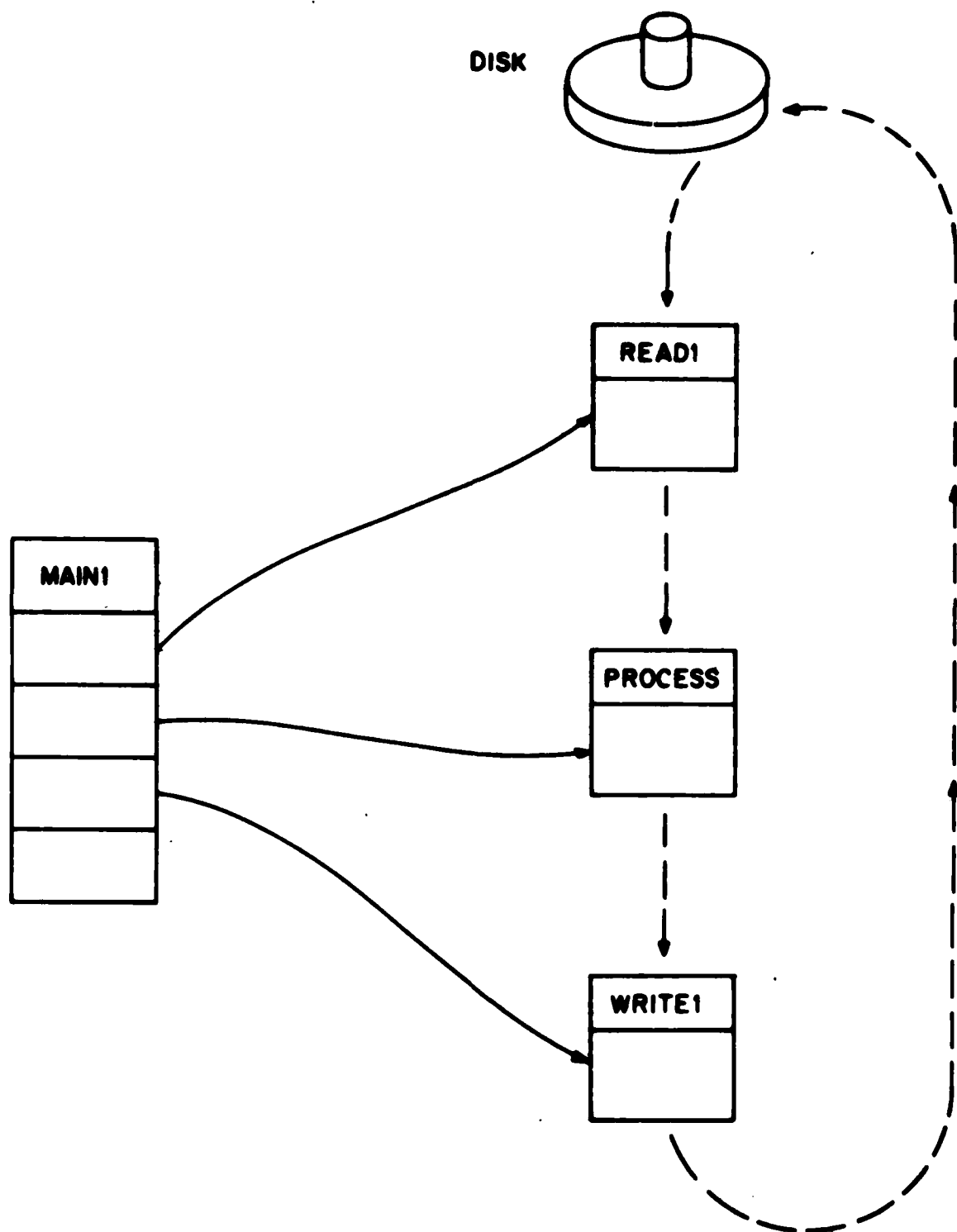


Figure 7. Structure of the preprocessing software.

```

0025      DO 30 L=1,2
0026      CALL RESUME(READ1)
0027      CALL RESUME(WRITE1)
0028      CALL RESUME(ROCE1)
0029      CALL WAITFR(38)
0030      CALL WAITFR(37)
0031      CALL WAITFR(36)
0032      CALL CLREF(38)
0033      CALL CLREF(37)
0034      CALL CLREF(36)
0035  30    CONTINUE
0036      CALL RESUME(WRITE1)
0037      CALL RESUME(PROCE1)
0038      CALL WAITFR(38)
0039      CALL WAITFR(37)
0040      CALL CLREF(38)
0041      CALL CLREF(37)
0042      CALL SETEF(40)
0043      CALL RESUME(WRITE1)
0044      CALL WAITFR(38)
0045      CALL REQUES(ARROWS)

```

READ1 has access to a common global area (line 8) where it reads in the data to be processed. After initializing (lines 19 - 20) it suspends itself until activated. When activated it places the last six rows it has read in the first six positions of the buffer (lines 25 - 27) to be processed. It then reads 32 rows and stores the last six into STORE. The bookkeeping for the change of buffers is done (lines 32 - 34). MAIN1 is signalled that READ1 is finished by the call to SETEF and there is a jump to 21 which suspends READ1.

```

0001      PROGRAM READ1
0002      INTEGER*2 SWITCH,POINT,WAKEUP,FINISH,OFFSET
0003      INTEGER*2 STORE(-383:0)
0004      INTEGER*2 INTIN(-383:2048,2),INTOUT(-383:2048,2)
0005      LOGICAL*1 INBUFF(128,-5:32,2) : INPUT BUFFER
0006      LOGICAL*1 OUBUFF(128,-5:32,2) : OUTPUT BUFFER
0007      LOGICAL*1 B10(25)
0008      COMMON /DTA/INBUFF,OUBUFF
0009      EQUIVALENCE (INTIN(-383,1),INBUFF(1,-5,1))
0010      EQUIVALENCE (OUBUFF(1,-5,1),INTOUT(-383,1))
0011      DATA WAKEUP,FINISH/33,36/

0019      POINT=1
0020      SWITCH=2
0021      CALL CLREF(40)

      C
0022      CALL SETEF(41)
0023  20    CALL SUSPND
0024      READ(10,POINT)(INTIN(J,SWITCH),J=1,2048)
0025      DO 15 I=-383,0
0026      INTIN(I,SWITCH)= STORE(I)
0027  15    CONTINUE

```

```

0028      DO 16 I=-383,0
0029      STORE(I)= INTIN( 2048+I,SWITCH)
0030      16  CONTINUE
0031      30  CONTINUE
0032      INTER=SWITCH
0033      IF(INTER.EQ.1) SWITCH=2
0034      IF(INTER.EQ.2) SWITCH=1
0035      IF(POINT,EQ.5) GOTO 40
0036      CALL SETEF(FINISH)
0037      GOTO 20 : GO WAIT TILL AWOKEN
0038      40  CALL SETEF(39)
0039      CALL SETEF(FINISH)
0040      END

```

The next step after the files have been read is to do the averaging and edge detection. Again PROCES has access to the global common area DTA. It initializes itself and then suspends itself and waits for MAIN1 to activate it when needed. The processing (lines 13, 14, 15) consists of an averaging operation, an edge detection (MOMENT) and another averaging. The processing is done from INBUF to OUTBUF (AVG), OUTBUF to INBUF (MOMENT), and then INBUF to OUTBUF. The bookkeeping to allow alternate buffers to be processed is then done; the program suspends itself and then waits for the next call.

```

0001      PROGRAM PROCESS
0002      INTEGER*2 SWITCH,POINT,WAKEUP,AVERAG,FINAVG
0003      REAL*4 M1,M2,MX,MY
0004      LOGICAL*1 INBUF(128,-5:32,2) : INPUT BUFFERS
0005      LOGICAL*1 OUTBUF(128,-5:32,2) : OUTPUT BUFFERS
0006      COMMON /DTA/ INBUF,OUTBUF
0007      DATA WAKEUP,FINAVG/34,37/
0008      SWITCH = 2
0009      KOUNT=0
0010      CALL CLREF(WAKEUP)
0011      20  CALL SUSPND
0012      KOUNT=KOUNT+1
0013      CALL AVERAG(SWITCH,31)
0014      CALL MOMENT(SWITCH,29)
0015      CALL AVERAG(SWITCH,27)
0016      CALL SETEF(FINAUG)
0017      INTER=SWITCH
0018      IF(INTER.EQ.1) SWITCH=2
0019      IF(INTER.EQ.2) SWITCH=1
0020      IF(KOUNT.EQ.4) GOTO 21
0021      GOTO 20
0022      21  CONTINUE
0023      END

```

```

0001      SURROUTINE AVERAG(SWITCH,L)
0002      LOGICAL*1 INBUF(128,-5:32,2),OUTBUF(128,-5:32,2)
0003      INTEGER*2 R1,R2,R3
0004      INTEGER*2 SWITCH,L
0005      COMMON /DTA/ INBUF,OUTBUF
0006      DO 10 J=-4,L
0007      DO 20 I=1,127
0008      R1=R2
0009      R2=R3
0010      I0 = INBUF(I+1,J-1,SWITCH).AND.255
0011      I1 = INBUF(I+1,J,SWITCH).AND.255
0012      I2 = INBUF(I+1,J+1,SWITCH).AND.255
0013      R3 = (I0+I1+I2)/3
0014      IAVG= (R1+R2+R3)/3
0015      OUTBUF(I,J-1,SWITCH)=IAVG.AND.255
0016      20 CONTINUE
0017      10 CONTINUE
0018      RETURN
0019      END

0001      SUBROUTINE MOMENT(SWITCH,L)
0002      LOGICAL*1 INBUF(128,-5:32,2),OUTBUF(128,-5:32,2)
0003      INTEGER*2 SWITCH
0004      COMMON /DTA/ INBUF,OUTBUF
0005      DO 30 J=-4,L
0006      DO 40 I=1,127
0007      I0=I3
0008      I1=I4
0009      I2=I5
0010      I3=I6
0011      I4=I7
0012      I5=I8
0013      I6=OUTBUF(I+1,J-1,SWITCH).AND.255
0014      I7=OUTBUF(I+1,J,SWITCH).AND.255
0015      I8=OUTBUF(I+1,J+1,SWITCH).AND.255
0016      XM=FLOAT(5*(I0-I8)+4*(I1+I3-I5-I7))
0017      YM=FLOAT(5*(I6-I2)+4*(I3+I7-I1-I5))
0018      M=SQRT(XM**2+YM**2)
0019      INBUF(I,J-1,SWITCH)=M.AND.255
0020      40 CONTINUE
0021      30 CONTINUE
0022      RETURN
0023      END

```

The average that is done is an unweighted average. The edge detector used is a moment operator which has been shown to perform well in the presence of noise. The next program that is called is WRITE1. The data structure here are the same as those used for READ1 with the same global common area being used. It also suspends itself and waits to be activated.

```

0001      PROGRAM WRITEL
0002      INTEGER*2 SWITCH,POINT,WKEUP,FINWRI,OFFSET
0003      INTEGER*2 INTOUT(-383:2048,2)
0004      LOGICAL*1 INBUF(128,-5:32,2)  !  INPUT BUFFERS
0005      LOGICAL*1 OUTBUF(128 -5:32,2)  .  OUTPUT BUFFERS
0006      LOGICAL*1 B10(25), CHARAC
0007      COMMON /DTA/INBUF,OUTBUF
0008      EQUIVALENCE (INTOUT(-383,1),OUTBUF(1,-5,1))
0009      DATA WAKEUP, FINWRI/35,38/

0018      POINT=1
0019      SWITCH=2
      C
0020      CALL CLREF(WAKEUP)
0021      CALL SETEF(42)
0022      20  CALL SUSPND
0023      WRITE(11'POINT')(INTOUT(J,SWITCH),J=-383,1664)
0024      30  CONTINUE
      C
0025      INTER=SWITCH
0026      IF(INTER.EQ.1) SWITCH=2
0027      IF(INTER.EQ.2) SWITCH=1
0028      CALL READEF(40,LCODE)
0029      IF(LCODE.EQ.2) GOTO 46
0030      CALL SETEF(FINWRI)
0031      GOTO 20
0032      46  CONTINUE
0033      INBUF(1,-5,1) = CHARAC
0034      DO 47 I=1,25
0035      INBUF(I+1,-5,1)= B10(I)
0036      47  CONTINUE
0037      CALL SETEF(FINWRI)
0038      END

```

A model for a hardware realization of this software is given in Figure 8. Here each of the circles would be a CPU together with some local memory. They would be passive and controlled by a CPU,MAIN. The squares would correspond to buffers accessed by CPU's as indicated. There are standard hardware methods, such as interrupts and flags, that can be used for the synchronization which is done in the software model.

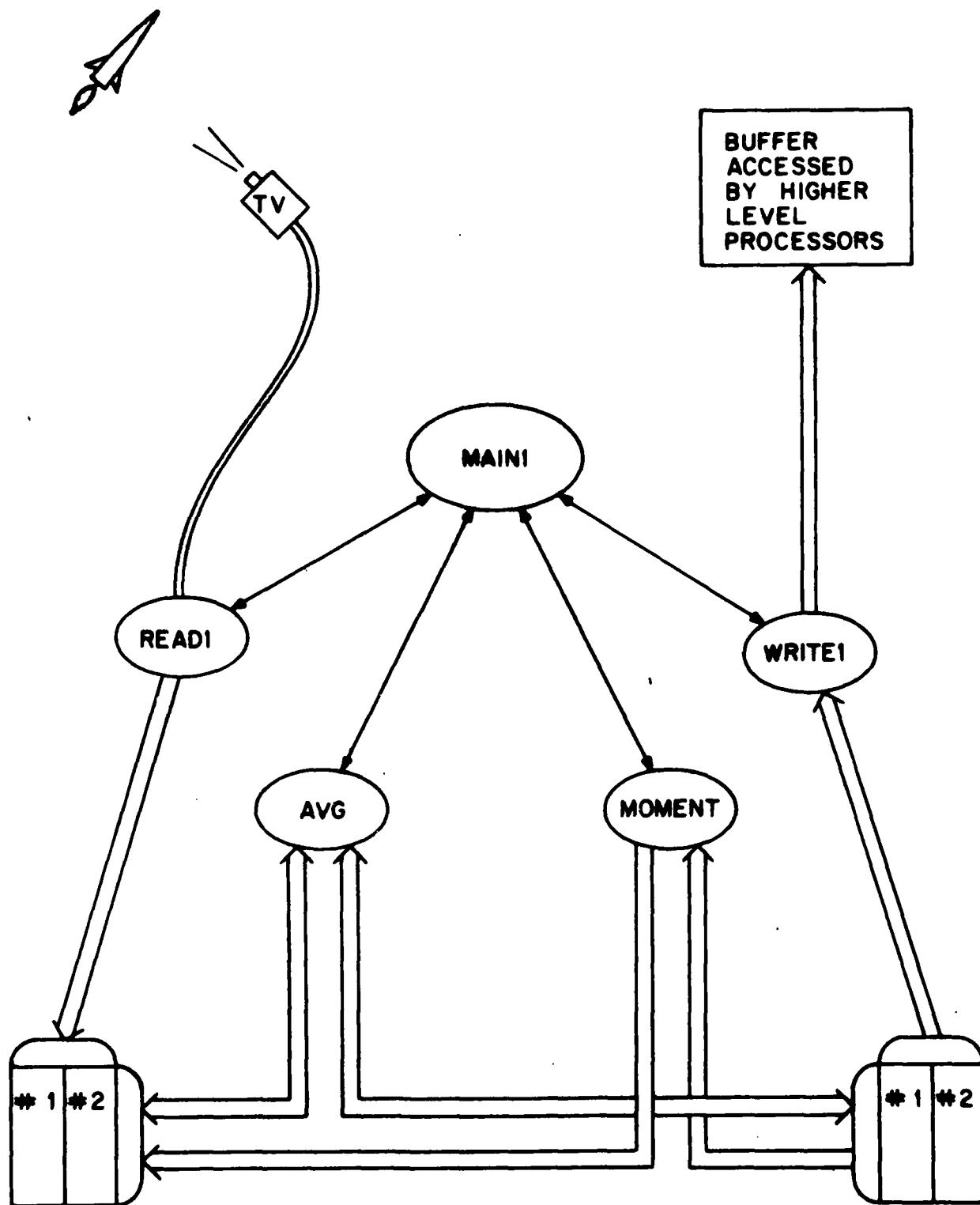


Figure 8. Generic hardware model for preprocessing hardware.

CONSTRUCTION OF A CONTOUR FROM THE MOMENT FILE

After the moment file PIC is created a threshold "T" must first be found such that a contour of the target will be included in the set.

$$\{ \text{PIC}(i,j) : i=1..n, j=1..N, \text{PIC}(i,j) < T \}$$

To find τ we first compute the histogram of the moment file created. A number p needs to have been chosen beforehand which represents the percentage of the scene points which are target points. The first point for which

$$\int_x^{255} \text{histo} > p$$

is found and used as the value of τ . It has been found that $p=15$ works well in cases where the target is a small part of the scene and $p=25$ does well when the target is a large percentage of the scene. In Figure 9 there are four originals that will be reduced to a contour. The result of preprocessing this data is in Figure 10. The problem now is to find a τ such that the target will be separated from the background. If we look at the raw histograms (Figure 11I) we can, in some cases, guess at where the threshold should be chosen, assuming that there is one distribution for the target and another for the background. The background distribution is centered about the maximum of the histogram while the target distribution is part of the tail of the histogram. Thus it is reasonable to suppose that the target points constitute a certain percentage of the points to the right of some value. Experiments have shown that the proper value for this percentage is between 15 and 25, depending on the size of the target. Figure 11b is a figure found from 11a by graphing

$$\text{sum}_i(x) = \int_x^{255} \text{histo}_i$$

for each histogram of 11(I). From this graph we see that, as the contrast decreases, the threshold to be chosen decreases, a procedure that agrees with our intuition. We can also see that, when the target size is large, the graph is radically different than when the target is small. Using 11(II) and $p=23$, we obtain the contours of Figure 12. The computer is set to threshold at $p=23$, the contour appears on the screen; and the operator can reject this contour and request a new one based on an operator supplied value for p . One choice of p does not always produce closed contours of the target; and this is why operator intervention is required at this point. As this system stands now, p is set by the operator on the initial frame and used for subsequent frames until the operator intervenes.

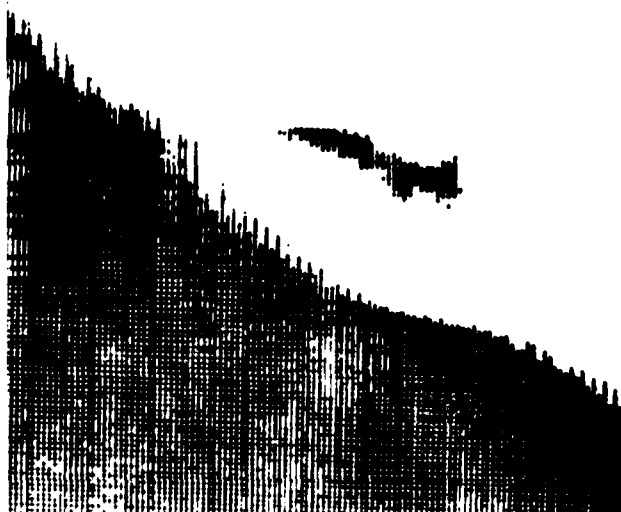


Figure 9a. F102 flying by a mountain.



Figure 9b. F102 flying in front of a mountain.



Figure 9c. Hawk missile



Figure 9d. F102, sunspots on wings and nose.



Figure 10a. Moment file of F102
flying by a mountain.



Figure 10b. Moment file of F102 flying
in front of a mountain.



Figure 10c. Moment file of Hawk
missile.



Figure 10d. Moment file of F102, sun-
spots on wings and nose.

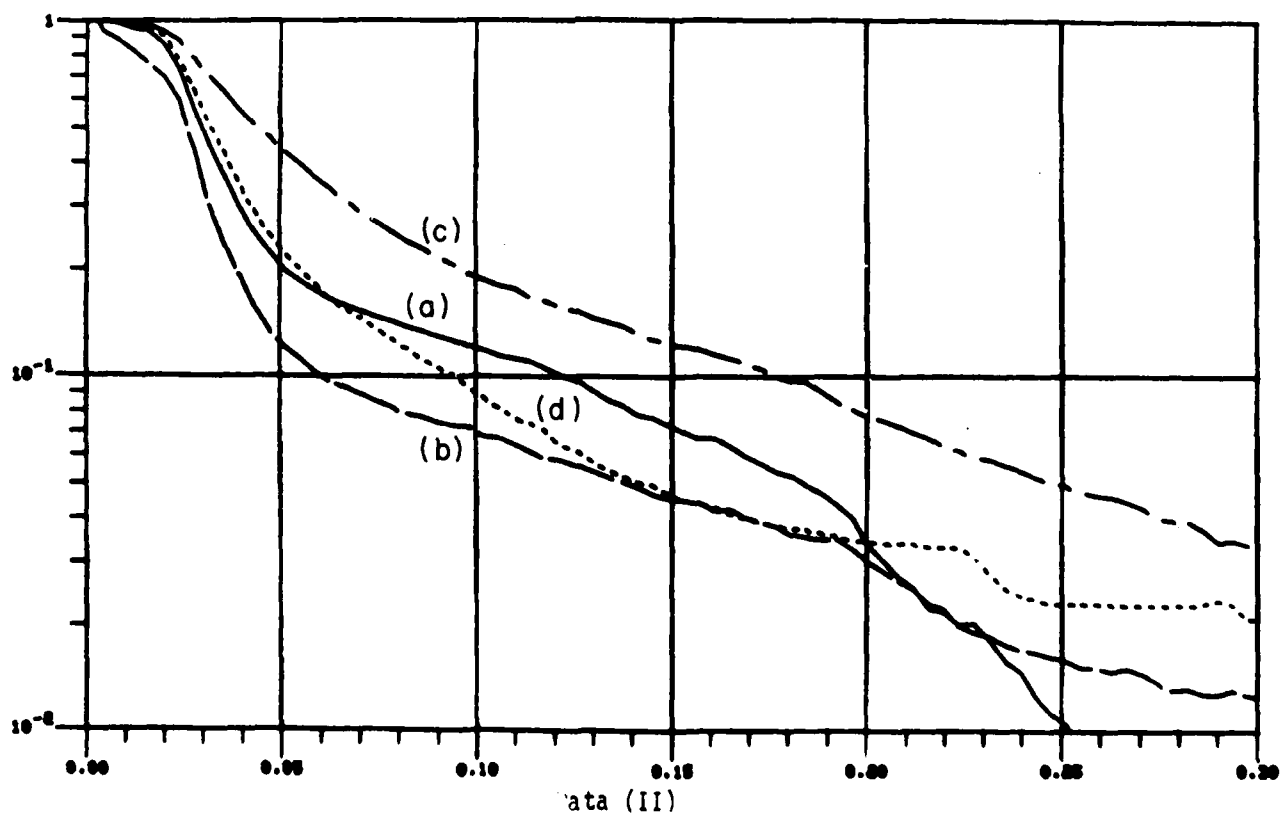
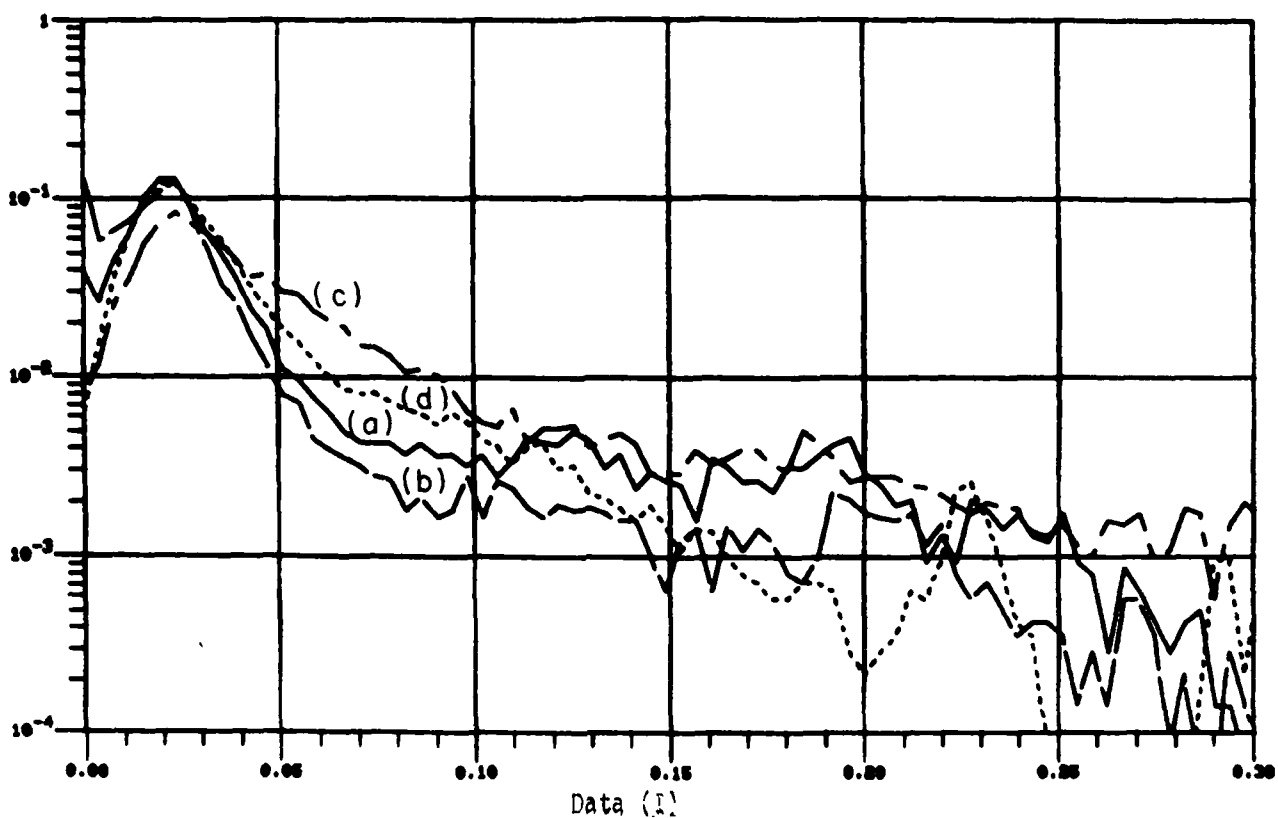


Figure 11. Raw histogram Data (I) from moment file. Integrated Data (II) from raw histograms.



Figure 12a. Contour found from Moment file (Fig. 10a.)



Figure 12b. Contour found from Moment file (Fig. 10b).



Figure 12c. Contour found from Moment file (Fig. 10c).



Figure 12d. Contour found from Moment file (Fig. 10d).

Once the value of T has been determined so that the target lies in the set

$$\{ \text{PIC}(i,j) : i=1 \dots n, j=1 \dots m, \text{PIC}(i,j) < T \}$$

We are in a position to begin making the contours from which the targets will be extracted. Again since these programs are intended to be used as models for a future hardware implementation the processing follows a sequential order. That is, the rows are processed from 1 to n with the original gray values used just once in the processing. It is commonly known that to construct a fast contour plotting algorithm the Freeman code must be dropped as soon as possible. We use the Freeman code to record the direction as we go along but convert to a polygonal representation as soon as it is possible, as when there occur two consecutive Freeman codes with the same direction. This is the first step in the contour-forming process and results in a set of line segments which are specified by their endpoints. The next step is joining the elements into contours.

The program that performs the operations described above is the program ARROW. First of all, six rows to be processed are read into AMMOUN(128,6). One pair of rows is processed at a time with the results of the processing being put into STOR(128,2) (lines 52 - 63). The assignment of directions begins by -- first, thresholding two rows of AMMOUN, with a point being assigned a zero if the average of a two-by-two neighborhood is greater than t and a one if it is less than t (lines 59 - 61). Beforehand STOR(128,1) is first overwritten by the last row processed which had been put in STOR(128,2) (lines 65 - 66).

```

0001      PROGRAM ARROW
0002      INTEGER*2 MAG(384),ANG(384),CENTER,THRSHM,X
0003      INTEGER*2 U,V,POINT3,AO,A1,A2,A3
0004      INTEGER*2 POINT1,POINT2,VAR,SUM,SIGN
0005      INTEGER*2 h0,H1,H2,H3,STOR(128,2),AVERAG,THRESH
0006      LOGICAL*1 Y,ANS,INV,FLAG
0007      LOGICAL*1 AMMOUN(128,6),ANGLE(128,6),NEW(128,5)
0008      LOGICAL*1 B10(25),NAMETE(26)
0009      REAL*4 K1,SEGMN
0010      INTEGER*2 LINES(500,3)COORDI(1500,6)
0011      INTEGER*2 POINSI
0012      INTEGER*2 POINTL,POINTH,POINTT,COL
0013      COMMON INV,FLAG,SIGN
0014      COMMON LINES,POINTL,POINTH,POINTT,COL,I28
0015      COMMON /DTA/COORDI
0016      EQUIVALENCE (MAG,AMMOUN)
0017      EQUIVALENCE (NAMETE,COORDI)
0018      EQUIVALENCE (ANG,ANGLE)
0019      DATA Y/89/
0020      DATA SEGMN/6RSEGMN/
0021      DATA IJ/0/
0022      DATA AK/1.4111764/
0023      CALL ERRSET(37,.TRUE.,.FALSE.,.FALSE.,.FALSE.,31)
0024      I=NAMETE(1)
0025      DO 999 M=1,25
0026      B10(M) = NAMETE(M+1)

```

```

0027 999 CONTINUE
0028 CALL ASSIGN(7,B10,I)
0029 DO 947 L=1,20
0030 COORDI(L,1)=0
0031 COORDI(L,2)=0
0032 COORDI(L,3)=0
0033 COORDI(L,4)=0
0034 COORDI(L,5)=0
0035 COORDI(L,6)=0
0036 947 CONTINUE
0037 POINSI=1
0038 DEFINE FILE 7(0,64,U,POINT2)
0039 POINT2=1
0040 CALL DEVIAT(THRESH,POINT2)
0041 POINT2=1
0042 IJ=0
0043 POINTT=1
0044 POINTL=0
0045 POINTH=1
0046 338 CONTINUE
0047 DO 150 K000=1,25
0048 DO 140 KO=0,5
0049 READ(7'POINT2,END=77) (MAG(I),I=1+KO*64,64+KO*64)
0050 140 CONTINUE
0051 77 CONTINUE
0052 DO 10 J=1,5
0053 DO 5 I=1,127
0054 H2=H0
0055 H3=H1
0056 LCONS=I+1
0057 H0=AMMOUN(LCONS,J).AND.255
0058 H1=AMMOUN(LCONS,J+1).AND.255
0059 AVERAG=(H0+H1+H2+H3)/4
0060 STOR(I,2)=0
0061 IF(AVERAG.LT.THRESH) STOR(I,2)=1
0062 14 CONTINUE
0063 5 CONTINUE
0064 CALL OUT(STOR,J+IJ)
0065 DO 756 IND=1,128
0066 STOR(IND,1)=STOR(IND,2)
0067 756 CONTINUE
0068 10 CONTINUE
0069 700 CONTINUE
0070 POINT2=POINT2-1
0071 IJ=IJ+5
0072 150 CONTINUE
0073 339 FORMAT(32X,15(X,14),/)
0074 341 CONTINUE
0075 CALL CLOSE(7)
0076 COORDI(1,6)=POINTH
0077 CALL CHANGE
0078 CALL REQUES(SEGMN)
0079 991 CONTINUE
0080 END

```

When the Subroutine OUT is called it uses STORE to generate Freeman-Code directions for a row and three such rows are stored in the array NUMBER(128,3).

To obtain a Freeman-Code, the patterns of Figure 13 are assigned the indicated values by the subroutine OUT and stored in NUMBER(128,3). All other patterns are assigned a -1.

```

0001      SUBROUTINE OUT(STORE,J)
0002      INTEGER*2 STORE(128,2),NUMBER(128,3)
0003      REAL*4 ANGLES(15)
0004      REAL*4 THETA
0005      LOGICAL*1 FLAG1,FLAG2,FLAG
0006      DATA ANGLES/-1.,7.,6.,3.,4.,-1.,5.,1.,-1.,0.,-1.,2.,3*-1./
0007      DATA FLAG/.FALSE./
0008      IF(J.GT.3)CALL LOGIC(NUMBER,J)
0009      CALL READEP(15,LCODE)
0010      IF(LCODE.EQ.2) RETURN
0011      DO 10 I=1,127
0012      NUMBER(I,1)=NUMBER(I,2)
0013      NUMBER(I,2)=NUMBER(I,3)
0014 10     CONTINUE
0015      DO 20 I=1,127
0016      INDEX=STORE(I,1)*2**3+STORE(I,2)*2**2+STORE(I+1,1)*2+ STORE(I+1,2)
0017      IF(INDEX.EQ.0) INDEX=15
0018      NUMBER(I,3)=IINT(ANGLES(INDEX))
0019 20     CONTINUE
0020      RETURN
0021      END

```

The next step is the linking of directions which are the same, and appear sequentially in a three-by-three window. Two predicates are used in controlling the statements to be executed. These are

P = The element of NUMBER is a continuation of a segment of the same direction.

Q = The element of NUMBER being checked is continued by a segment of the same direction.

The cases where $P=.true.$ and $Q=.true.$ are illustrated in Figures 14a, b. The possible predicates and the actions taken when the predicates are true are shown in Figure 14c.

a. $P \wedge Q$

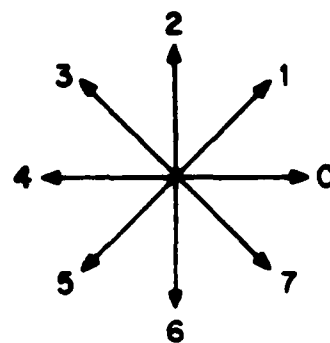
Put the segment number of the line that the element of NUMBER continues into first 15 bits of NUMBER(i,2). This segment number is extracted from the first 15 bits that NUMBER(i,2) continues.

b. $\bar{P} \wedge Q$

In this case a new segment needs to be started. The starting row and columns are stored in the array LINES. POINTL contains the current

I , 1	I+1,1
I , 2	I+1,2

STOR



1	1
0	0



1	0
0	0



1	0
1	0



0	0
1	0



0	0
1	1



0	0
0	1



0	1
0	1



0	1
0	0



Figure 13. Two-dimensional number patterns and their assigned geometrical directions.

segment number and `LINES(POINTL,1)` sets the row number, `LINES(POINTL,2)` the column number and `LINES(POINTL,3)` the direction.

c. P and \bar{Q}

This happens when a line segment of the same direction is terminated. The action taken in this case is to store the beginning coordinates ending coordinates, and direction of the line segment in `COORDI`. The beginning point of the segment is obtained by first stripping the first 15 bits off `NUMBER(i,2)` and accessing the entry of `LINES` which corresponds to this number. This gives the starting point of the segment, while the final point is gotten from the current row and column coordinates.

d. \bar{P} and \bar{Q}

It is an isolated direction and thus it is stored directly into `COORDI`.

Subroutines `LOGIC` and `GUARDS` look at the direction numbers in `NUMBER` and link those arrows that occur sequentially in the same direction. The arrays used to do the bookkeeping at this stage are `LINES(,)` and `COORDI(,)` with the final results being stored in `COORDI(,)`. Long polygonal segments are constructed by tracking along consistent joins of these line segments at each point, checking for possible continuation of each segment.

The subroutine `LOGIC` drives the programs which produce the pseudo Freeman code and do the bookkeeping functions. The data from which it computes line segments is in `NUMBER(I28,3)` and it consists of the Freeman codes generated from the last three rows processed. The first fifteen bits of `NUMBER` are used to store the segment number of a particular entry.

`LOGIC` processes a row in the do loops of line fourteen to twenty-four. This loop begins by looking to see if the element `NUMBER(I28,2)` is a possible edge element, and if it finds that the element equals "-1" it looks at a new element since the non-edge elements have been assigned a "-1." If it is a possible edge element it extracts the segment number and calls `GUARDS` to compute the values of P and Q . What is left to do now are the actions which correspond to different values of P and Q this is done in lines 20 through 23.

```

0001      SUBROUTINE LOGIC(NUMBER,J)
0002      INTEGER*2 NUMBER(128,3)
0003      INTEGER*2 LINES(500,3),COORDI(1500,6)
0004      LOGICAL*1 P ! IF P=TRUE THEN ARROW IS A CONTINUATION
0005      LOGICAL*1 Q ! IF Q=TRUE THEN ARROW IS CONTINUED
0006      INTEGER*2 ROW,COL,POINTT,POINTH,POINTL,I
0007      LOGICAL*1 INV,FLAG
0008      INTEGER*2 SIGN,POINSI
0009      COMMON INV,FLAG,SIGN
0010      COMMON LINES,POINTL,POINTH,POINTT,COL,I28
0011      COMMON /DTA/ COORDI
0012      IF(J.EQ.1) POINTL=1
0013      IF(J.EQ.1) POINTH=1
0014      DO 10 I28=2,127

```

```

0015      IF (NUMBER(I28,2).EQ.-1)GOTO 10
0016      COL=NUMBER(I28,2).AND."177770
0017      CALL GUARDS(NUMBER,P,Q,I28)
0018      ROW=NUMBER(I28,2).AND.7           !DIRECTION CODE
0019      COL=(NUMBER(I28,2).AND."177770)/8 !COMPONENT NUMBER
0020      IF (P.AND..NOT.Q) CALL DUMP(J)
0021      IF ((.NOT.P).AND.Q) CALL BEGIN(ROW,J)
0022      IF ((.NOT.P).AND.Q) NUMBER(I28,2)=NUMBER(I28,2).OR.(POINTL*8)
0023      IF ((.NOT.P).AND.(.NOT.Q)) CALL SINGLE(NUMBER,J,ROW)
0024  10   CONTINUE
0025      RETURN
0026      END

```

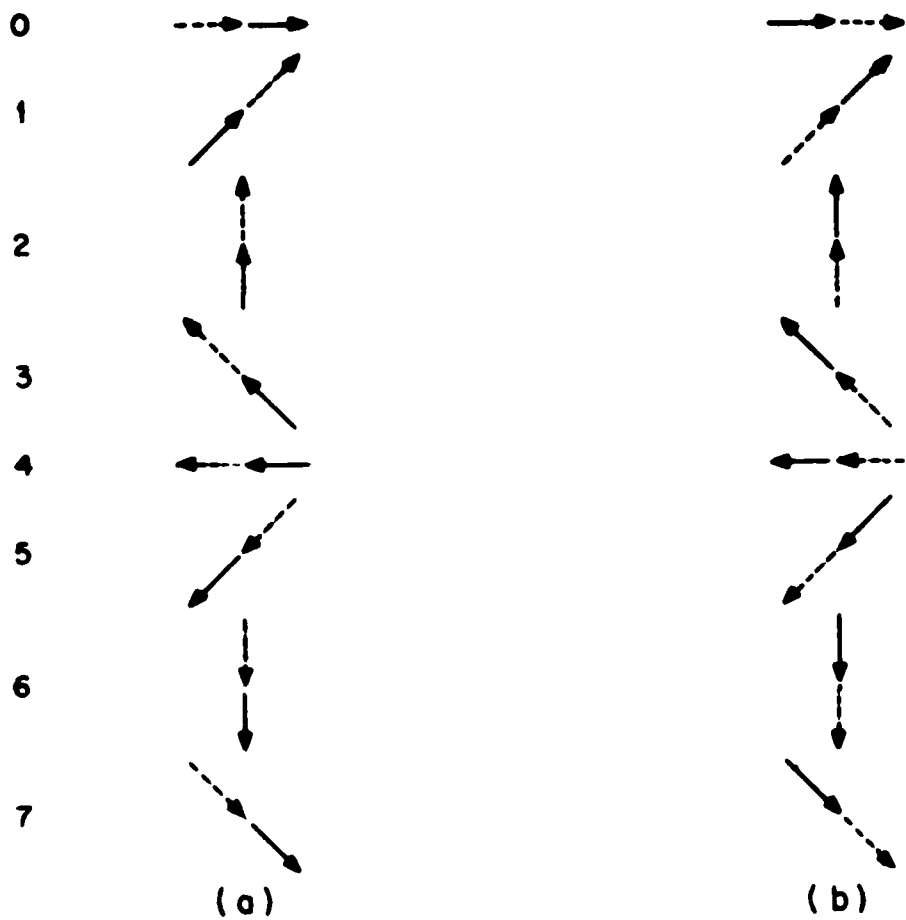
LOGIC computes the values of P and Q by using the subroutine GUARDS. GUARDS looks in a three-by-three neighborhood of STATES(I28,3) and computes the value of P and Q for STATES(I,2). It checks, as in Figure 14, for the appropriate values of P, lines 12 - 57, and then goes on to compute Q, line 60 - end. This processing is done for the entire file with the final results being stored into COORDI. COORDI(1500,6) is now sorted on the row coordinates of its elements. The format of the elements, that are stored in COORDI, is also changed so that the data is now

CODE,BEGIN(j),BEGIN(i),DEL

```

0001      SUBROUTINE GUARDS(STATES,P,Q,I)
0002      INTEGER*2 I,STATES(128,3),SEGMENT,CODE
0003      INTEGER*2 CHOICP,CHOICQ
0004      LOGICAL*1 P,Q,L
0005      P=.FALSE.
0006      Q=.FALSE.
0007      L=.FALSE.
0008      CODE=(STATES(I,2).AND.7)
0009      INDEX=CODE+1
0010      SEGMENT=0
0011      CHOICP=-1
0012      GOTO(10,20,30,40,50,60,70,80)INDEX
0013  10   F((STATES(I-1,2).AND.7).EQ.0)CHOICP=STATES(I-1,2)
0014  0014 IF(CHOICP,EQ.-1) GOTO 85
0015      SEGMENT=CHOICP.AND."177770
0016      IF (SEGMENT,EQ.0)GOTO 85
0017      P=.TRUE.
0018      STATES(I,2)=SEGMENT.OR.CODE
0019      GOTO 85
0020  20   IF((STATES(I+1,1).AND.7).EQ.1)SEGMENT=STATES(I+1,1).AND."177770
0021      IF (SEGMENT,EQ.0) GOTO 85
0022      P=.TRUE.
0023      STATES(I,2)=SEGMENT.OR.CODE
0024      GOTO 85
0025  30   IF((STATES(I,1).AND."177770.EQ.2)SEGMENT=STATES(I,1).AND."177770
0026      IF (SEGMENT,EQ.0) GOTO 85
0027      P=.TRUE.

```



$\underline{P \text{ and } Q}$

$\overline{P \text{ and } Q}$

$\underline{P \text{ and } \overline{Q}}$

$\overline{P \text{ and } \overline{Q}}$

Figure 14. Examples of geometric relations for various predicates, the geometric situation when (a) $P = \text{true}$, and (b) $Q = \text{true}$.

```

0028      STATES(I,2)=SEGMEN.OR.CODE
0029      GOTO 85
0030  40    IF((STATES(I-1,1).AND.7).EQ.3) SETMEN=STATES(I-1,1).AND."177770
0031      IF(SEGMEN.EQ.0) GOTO 85
0032      P=.TRUE.
0033      STATES(I,2)=SEGMEN.OR.CODE
0034      GOTO 85
0035  50    IF((STATES(I-1,2).AND.7).EQ.4) SEGMEN=STATES(I-1,2).AND."177770
0036      IF(SEGMEN.EQ.0) GOTO 85
0037      P=.TRUE.
0038      STATES(I,2)=SEGMEN.OR.CODE
0039      GOTO 85
0040  60    IF((STATES(I+1,1).AND.7).EQ.5) SEGMEN=STATES(I+1,1).AND."177770
0041      IF(SEGMEN.EQ.0) GOTO 85
0042      P=.TRUE.
0043      STATES(I,2)=SEGMEN.OR.CODE
0044      GOTO 85
0045  70    IF((STATES(I,1).AND.7).EQ.6) CHOICP=STATES(I,1)
0046      IF(CHOICP.EQ.-1) GOTO 85
0047      SEGMEN=CHOICP.AND."177770
0048      P=.TRUE.
0049      STATES(I,2)=SEGMEN.OR.CODE
0050      GOTO 85
0051  80    IF((STATES(I-1,1).AND.7).EQ.7) CHOICP=STATES(I-1,1)
0052      IF(CHOICP.EQ.-1) GOTO 85
0053      SEGMEN=CHOICP.AND."177770
0054      IF(SEGMEN.EQ.0) GOTO 85
0055      P=.TRUE.
0056      STATES(I,2)=SEGMEN.OR.CODE
0057      GOTO 85
0058  85    CHOICP=-1
0059      CHOICQ=-1
0060      GOTO(100,200,300,400,500,600,700,800) INDEX
0061  100   IF((STATES(I+1,2).AND.7).EQ.0) CHOICQ=STATES(I+1,2)
0062      IF(CHOICQ.EQ.-1) GOTO 850
0063      Q=.TRUE.
0064      GOTO 850
0065  200   IF((STATES(I-1,3).AND.7).EQ.1) CHOICQ=STATES(I-1,3)
0066      IF(CHOICQ.EQ.-1) GOTO 850
0067      Q=.TRUE.
0068      GOTO 850
0069  300   IF((STATES(I,3).AND.7).EQ.2) CHOICQ=STATES(I,3)
0070      IF(CHOICQ.EQ.-1) GOTO 850
0071      Q=.TRUE.
0072      GOTO 850
0073  400   IF((STATES(I+1,3).AND.7).EQ.3) CHOICQ=STATES(I+1,3)
0074      IF(CHOICQ.EQ.-1) GOTO 850
0075      Q=.TRUE.
0076      GOTO 850
0077  500   IF((STATES(I+1,2).AND.7).EQ.4) CHOICQ=STATES(I+1,2)
0078      IF(CHOICQ.EQ.-1) GOTO 850
0079      Q=.TRUE.

```

```

0081 600 IF((STATES(I-1,3).AND.7).EQ.5) CHOICQ=STATES(I-1,3)
0082 IF(CHOICQ.EQ.-1) GOTO 850
0083 Q=.TRUE.
0084 GOTO 850
0085 700 IF((STATES(I,3).AND.7).EQ.6) CHOICQ=STATES(I,3)
0086 IF(CHOICQ.EQ.-1) GOTO 850
0087 Q=.TRUE.
0088 GOTO 850
0089 IF((STATES(I+1,3).AND.7).EQ.7) CHOICQ=STATES(I+1,3)
0090 IF(CHOICQ.EQ.-1) GOTO 850
0091 Q=.TRUE.
0092 GOTO 850
0093 850 RETURN
0094 END

```

```

0001 SUBROUTINE BEGIN(DIRECT,J)
0002 INTEGER*2 LINES(500,3),COORDI(1500,6)
0003 INTEGER*2 COL,I,J,DIRECT,POINTL,POINTH,POINTT
0004 LOGICAL*1 INV,FLAG
0005 INTEGER*2 SIGN
0006 COMMON INV,FLAG,SIGN
0007 COMMON LINES,POINTL,POINTH,POINTT,COL,I28
0008 COMMON /DTA/COORDI
0009 POINTL=POINTL+1
0010 LINES(POINTL,1)=I28
0011 LINES(POINTL,2)=J
0012 LINES(POINTL,3)=DIRECT
0013 RETURN
0014 END

```

```

0001 SUBROUTINE DUMP(J)
0002 INTEGER*2 LINES(500,3),COORDI(1500,6)
0003 INTEGER*2 I,J,DIRECT,POINTL,POINTH,POINTT
0004 REAL*4 BEGINX,BEGINY,ENDX,ENDY
0005 INTEGER*2 SIGN,COL
0006 LOGICAL*1 INV,FLAG
0007 COMMON INV,FLAG,SIGN
0008 COMMON LINES,POINTL,POINTH,POINTT,COL,I28
0009 COMMON /DTA/COORDI
0010 DIRECT=LINES(COL,3)
0011 BEGINX=30+LINES(COL,1)*7
0012 BEGINY=775-LINES(COL,2)*7
0013 ENDX=30+(I28)*7
0014 ENDY=775-(J)*7
0015 GOTO(100,200,300,400,500,600,700,800) DIRECT+1
0016 100 ENDX=ENDX+7
0017 GOTO 1000
0018 200 BEGINX=BEGINX+7
0019 BEGINY=BEGINY+7
0020 GOTO 1000
0021 300 BEGINY=BEGINY+7
0022 GOTO 1000

```

```

0023 400 BEGINX=BEGINX-7
0024 BEGINY=BEGINY+7
0025 GOTO 1000
0026 BEGINX=BEGINX-7
0027 GOTO 1000
0028 600 ENDX=ENDX-7
0029 ENDY=ENDY-7
0030 GOTO 1000
0031 700 ENDY=ENDY-7
0032 GOTO 1000
0033 800 ENDX=ENDX+7
0034 ENDY=ENDY-7
0035 GOTO 1000
0036 1000 CONTINUE
0037 10 COORDI(POINTH,1)=-(BEGINY-775)/7
0038 COORDI(POINTH,2)=(BEGINX-30)/7
0039 COORDI(POINTH,3)=DIRECT
0040 20 COORDI(POINTH,4)=-(ENDY-775)/7
0041 COORDI(POINTH,5)=(ENDX-30)/7
0042 POINTH=POINTH+1
0043 RETURN
0044 END

0001 SUBROUTINE CHANGE
0002 INTEGER*2 COORDI(1500,6) ! INPUT DATA
0003 INTEGER*2 POINT ! AMOUNT OF DATA
0004 LOGICAL*1 FLAG,B10(25)
0005 INTEGER*2 LINES(500,3),POINTL,POINTH,POINTT
0006 INTEGER*2 COL,128,SIGN
0007 LOGICAL*1 INV
0008 COMMON INV,FLAG,SIGN
0009 COMMON LINES,POINTL,POINTH,POINTT,COL,128
0010 COMMON /DTA/COORDI
0011 POINT=0
0012 FLAG=.TRUE.
0013 CALL ERRSET(37,.TRUE.,.FALSE.,.FALSE.,.FALSE.,31)
0014 100 CONTINUE
0015 C COORDI FORMAT=CODE,TAIL(J),TAIL(I),DELTA
0016 DO 90 I=2,POINTH
0017 GOTO(10,20,20,20,30,40,40,40) COORDI(I,3)+1
0018 10 FLAG=COORDI(I,2).GE.COORDI(I,5)
0019 COORDI(I,6)=IABS(COORDI(I,2)-COORDI(I,5))
0020 IF(.NOT.FLAG)COORDI(I,4)=COORDI(I,1)
0021 IF(.NOT.FLAG)COORDI(I,5)=COORDI(I,2)
0022 GOTO 88
0023 20 FLAG=COORDI(I,1).GE.COORDI(I,4)
0024 COORDI(I,6)=IABS(COORDI(I,1)-COORDI(I,4))
0025 IF(FLAG) COORDI(I,4)=COORDI(I,1)
0026 IF(FLAG) COORDI(I,5)=COORDI(I,2)
0027 GOTO 88

```

```

0027 30 FLAG=COORDI(I,5).GE.COORDI(I,2)
0028 COORDI(I,6)=IABS(COORDI(I,2)-COORDI(I,5))
0029 IF(.NOT.FLAG) COORDI(I,4)=COORDI(I,1)
0030 IF(.NOT.FLAG) COORDI(I,5)=COORDI(I,2)
0031 GOTO 88
0032 40 FLAG=COORDI(I,4).GE.COORDI(I,1)
0033 COORDI(I,6)=IABS(COORDI(I,1)-COORDI(I,4))
0034 IF(FLAG) COORDI(I,4)=COORDI(I,1)
0035 IF(FLAG) COORDI(I,5)=COORDI(I,2)
0036 GOTO 88
0037 88 CONTINUE
0038 COORDI(I,1)=COORDI(I,3)
0039 COORDI(I,2)=COORDI(I,4)
0040 COORDI(I,3)=COORDI(I,5)
0041 COORDI(I,4)=COORDI(I,6)
0042 90 CONTINUE
0043 CALL SORT(POINTH,2)
0044 DO 199 I=3,POINTH-3
C WRITE(9,250) COORDI(I,1),COORDI(I,2),COORDI(I,3),COORDI(I,4)
0045 250 FORMAT(2X,I1,X,I3,X,I3,X,I3)
0046 199 CONTINUE
0047 RETURN
0048 END

0001 SUBROUTINE SORT(POINT,KEY)
0002 INTEGER*2 COORDI(1500,6) ! TRANSMITTED VIA COMMON
0003 INTEGER*2 POINT ! NUMBER OF ELEMENTS TO SORT
0004 INTEGER*2 KEY ! WHICH COLUMN TO SORT ON
0005 INTEGER*2 L,R,K,X(4) ! INTERMEDIATE LOCATIONS
0006 INTEGER*2 LINES(500,3)
0007 INTEGER*2 POINTL,POINTH,POINTT,COL,SIGN
0008 LOGICAL*1 FLAG,INV
0009 COMMON INV,FLAG,SIGN
0010 COMMON LINES,POINTL,POINTH,POINTT,COL,I28
0011 COMMON /DTA/COORDI
0012 L=2
0013 R=POINT
0014 1 CONTINUE ! RETURN POINT FOR OUTER REPEAT
0015 DO 2 J=R,L,-1
0016 IF(COORDI(J,KEY).GE.COORDI(J-1,KEY) )GOTO 2
0017 X(1)=COORDI(J-1,1)
0018 X(2)=COORDI(J-1,2)
0019 X(3)=COORDI(J-1,3)
0020 X(4)=COORDI(J-1,4)
0021 COORDI(J-1,1)=COORDI(J,1)
0022 COORDI(J-1,2)=COORDI(J,2)
0023 COORDI(J-1,3)=COORDI(J,3)
0024 COORDI(J-1,4)=COORDI(J,4)
0025 COORDI(J,1)=X(1)
0026 COORDI(J,2)=X(2)
0027 COORDI(J,3)=X(3)
0028 COORDI(J,4)=X(4)
0029 K=J
0030 2 CONTINUE

```



```

0031      L=K+1
0032      DO 3 J=L,R
0033      IF (COORDI(J,KEY).GE.COORDI(J-1,KEY)) GOTO 3
0034      X(1)=COORDI(J-1,1)
0035      X(2)=COORDI(J-1,2)
0036      X(3)=COORDI(J-1,3)
0037      X(4)=COORDI(J-1,4)
0038      COORDI(J-1,1)=COORDI(J,1)
0039      COORDI(J-1,2)=COORDI(J,2)
0040      COORDI(J-1,3)=COORDI(J,3)
0041      COORDI(J-1,4)=COORDI(J,4)
0042      COORDI(J,1)=X(1)
0043      COORDI(J,2)=X(2)
0044      COORDI(J,3)=X(3)
0045      COORDI(J,4)=X(4)
0046      K=J
0047 3     CONTINUE
0048      R=K-1
0049      IF (L.LE.R) GOTO 1
0050      END
0001      SUBROUTINE SINGLE(STATES,J,ROW)
0002      INTEGER*2 I,J,DIRECT
0003      INTEGER*2 LINES(500,3),COORDI(1500,6)
0004      LOGICAL*1 R,S,NR,NS,FLAG, INV
0005      INTEGER*2 SIGN,ROW,COL,POINTT,POINTH,POINTL,DIR
0006      INTEGER*2 CODE, SEGMENT,CHOICP,CHOICQ,POINSI
0007      REAL*4 BEGINX,BEGINY,ENDX,ENDY
0008      INTEGER*2 STATES(128,3)
0009      COMMON INV,FLAG,SIGN
0010      COMMON LINES,POINTL,POINTH,POINTT,COL,I28
0011      COMMON /DTA/COORDI
0012      CODE=STATES(I28,2).AND.7
0013      DIRECT=CODE
0014      BEGINX=30+(I28)*7
0015      BEGINY=775-(J)*7
0016      ENDX=30+(I28)*7
0017      ENDY=775-(J)*7
0018      GOTO(100,200,300,400,500,600,700,800) DIRECT+1
0019 100    ENDX=ENDX+7
0020        GOTO 1000
0021 200    BEGINX=BEGINX+7
0022        BEGINY=BEGINY+7
0023        GOTO 1000
0024 300    BEGINY=BEGINY+7
0025        GOTO 1000
0026 400    BEGINX=BEGINX-7
0027        BEGINY=BEGINY+7
0028        GOTO 1000
0029 500    BEGINX=BEGINX-7
0030        GOTO 1000
0031 600    ENDX=ENDX-7
0032        ENDY=ENDY-7
0033        GOTO 1000

```

```

0034    700    ENDY=ENDY-7
0035                GOTO 1000
0036    800    ENDX=ENDX+7
0037                ENDY=ENDY-7
0038                GOTO 1000
0039    1000   CONTINUE
0040                COORDI(POINTH,1)=-(BEGINY-775)/7
0041                COORDI(POINTH,2)=(BEGINX-30)/7
0042                COORDI(POINTH,3)=DIRECT
0043                COORDI(POINTH,4)=-(ENDY-775)/7
0044                COORDI(POINTH,5)=(ENDX-30)/7
0045                COORDI(POINTH,6)=DIRECT
0046                POINTH=POINTH+1
0047                RETURN
0048                END

```

The fifth coordinate will be used to place pointers that will give the next piece of a particular polygonal line, if there is one. At this point the scene has been reduced to a number of line segments of different length, each having one of eight possible directions. The next step is to link these by checking to see if there is a possible continuation of one segment by some other segment. Such a linking of segments is the function of the program SEGARR. To begin with, all elements of COORDI(*,5) are set equal to zero, after which a number of segments is built up in the following steps:

1. Look through COORDI(*,5), and if an entry is found equal to zero then proceed, or else stop.

2. Start a new segment by recording the location of the zero entry of COORDI(*,5) in SEGS.

3. Now look for an element in COORDI that satisfies linking criteria as given in Figure 16. If such an element is found, two different cases will be considered. Either it is a single element, or it is a segment (more than one element). The two alternative courses of action are:

- 'Segment' (a) Link-up data structures, as in Figure 15a, which results in the graphic operations (Figure 15c).

- 'Single' (b) Link-up data structures, as in Figure 15b, which results in the graphic operations (Figure 15d).

After these segments have been created the segment list is looked through, and if there is a consistent join of two segments whose distance apart is less than three units, then these are joined.

The program SEGARR links together the segments which are stored in the common area DTA. The number of elements in COORDI is passed via the sixth element of COORDI (lines 9 - 11). COORDI(*,5) will be used to store the pointers and they are all initialized to zero in lines 16 - 18. The line with label 85 is the beginning of the code which constructs the polygons from the long line segments. First, COORDI(*,5) is searched for a zero, i.e., a segment that

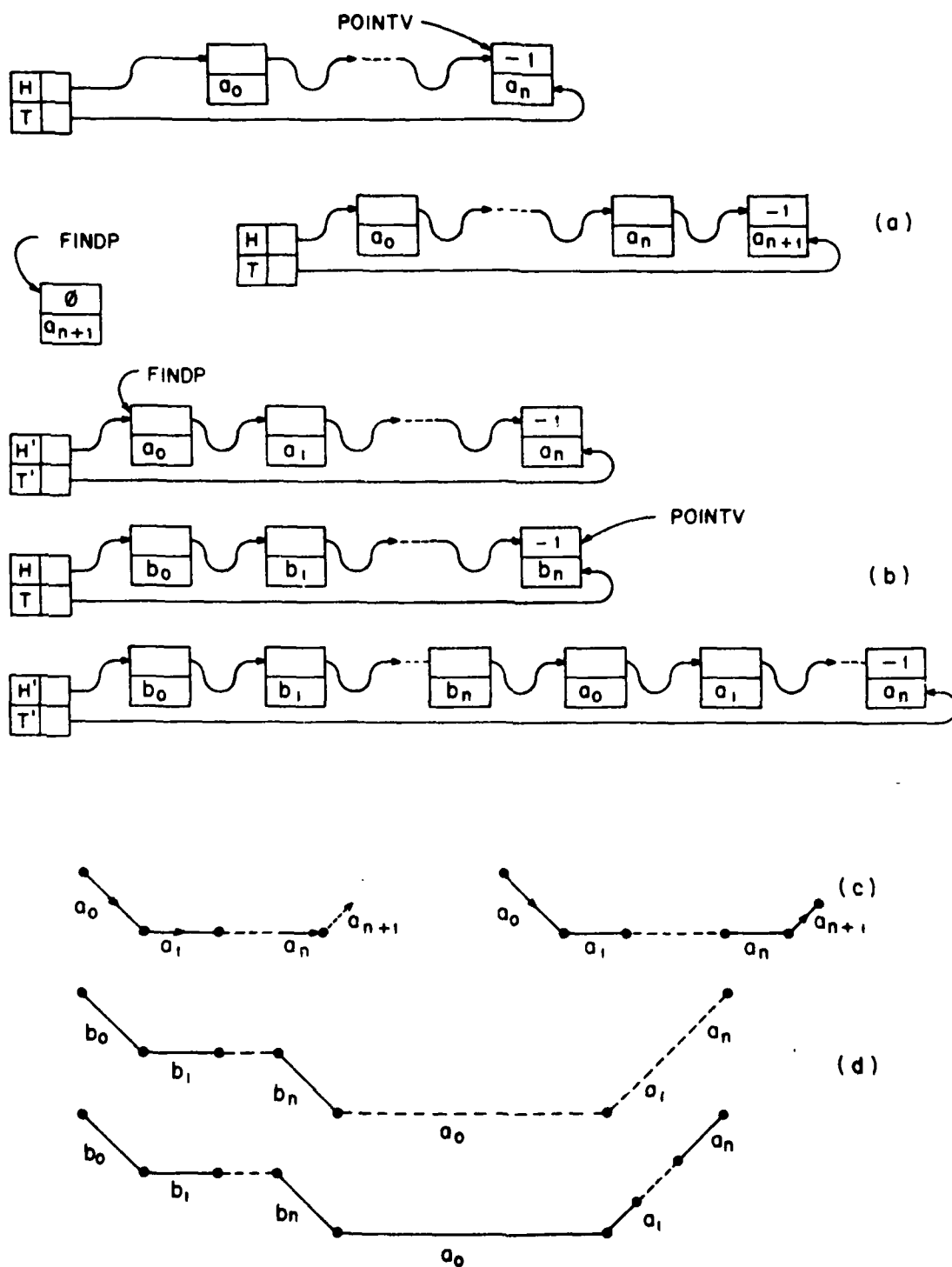


Figure 15. Data structures (a), (b) and operations with their corresponding (c), (d) geometric structures and geometric operations.

hasn't been used in the construction of a polygon. If it finds a zero, it jumps out of that loop and begins constructing the data structure which corresponds to a polygon (lines 20 - 23).

The pointer which keeps track of the numbers of polygons POINTS gets updated, and the index of the segment to be processed gets stored both in SEGS(POINTS,1) and SEGS(POINTS,2); also an "-1" gets stored in COORDI(I,5) to indicate the end of a polygon (lines 26 - 30). The call to WHERE computes the endpoint of the current segment being analyzed and stores it in the array SEARCH(3). SEARCH(1) containing the direction of the segment and SEARCH(2), SEARCH(3) the column and row coordinates (lines 33 - 34). To determine if there is a segment of COORDI that continues the segment at POINTV a call to FIND is made on line 34. This Subroutine returns an 0 in FINDP if there is no continuation. It points to the continuation of the segment being analyzed if one has been found that passes the test in FIND (Figure 16, Lines 19 - 86). If a continuation has been found and it has COORDI(FINDP,J) = 0 then it is one of the original long line segments and it is added to the list being constructed (lines 35 - 40). Alternatively it may be that there is a continuation of the element being tested but that this continuation is a segment. In this case the list which is being tested is added to the continuation list (Lines 42 - 46) with the corresponding list operations as in Figure 15. The resulting contour has many of the important segments in it but there are many gaps in the contours (Figure 17) which should be closed. One obvious method of closing these is to search through all the segments and join those that are less than a certain distance apart (Lines 52 - 77). This works fairly well, as Figure 18 shows.

```

0001      INTEGER*2 SEARCH(3)  !  !  CODE,HEAD(J),HEAD(I)
0002      INTEGER*2 COORDI(1500,6)  !  CODE,BEGIN(J),BEGIN(I),DEL,POINTER
0003      INTEGER*2 POINTH !  ALIAS FOR POINT
0004      INTEGER*2 SEGS(500,4)  !  BEGIN,END
0005      INTEGER*2 POINTV,POINTS,POINT,DI,DJ,FINDP
0006      INTEGER*2 MIN ,INDEX,H,T,Y(3)
0007      LOGICAL *1 BIO(25),FLAG,INTERN
0008      COMMON /DIS00/ SEGS,POINTS
0009      COMMON /DTA/COORDI
0010      CALL ERRSET(37,.TRUE.,.FALSE.,.FALSE.,.FALSE.,31)
0011      POINT=COORDI(1,6)
0012      ISAVE=1
0013      K1=0
0014      602  CONTINUE
0015          POINTS=0
0016          DO 90 I=2,POINT+3  !  SET ALL POINTERS TO ZERO
0017              COORDI(I,5)=0
0018          90  CONTINUE
0019          85  CONTINUE !  RETURN HERE TO BEGIN A SEGMENT
0020          DO 80 I=ISAVE+1,POINT+1  !  SEARCH FOR UNUSED ONES
0021              ISAVE=I
0022              IF(COORDI(I,5).EQ.0) GOTO 70
0023          80  CONTINUE

```

TEST1 = IN-12

TEST2 = JN-J2

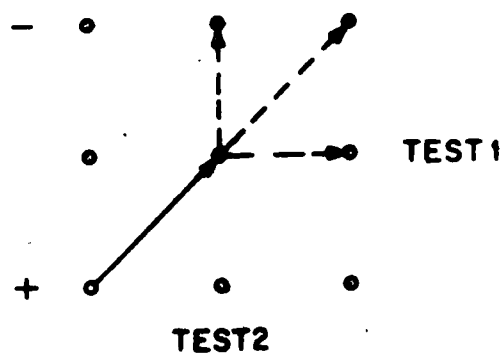
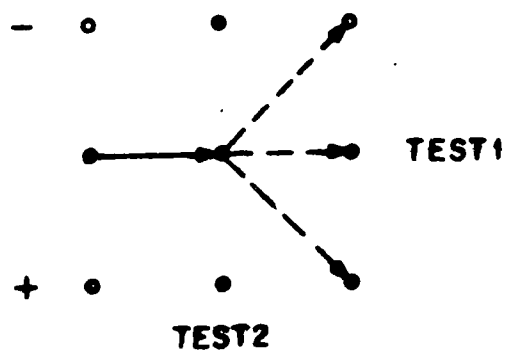


Figure 16. Geometric criteria for the continuation of a polygonal segment.

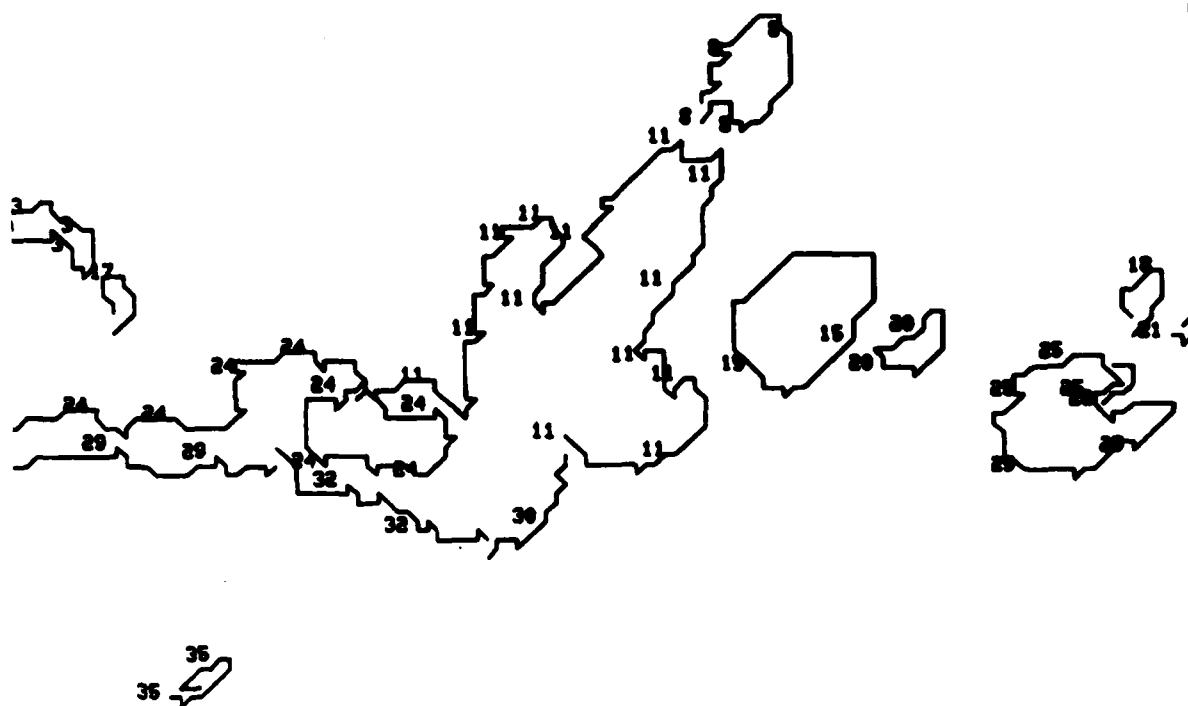


Figure. 17. Polygon before closing by distance measure.

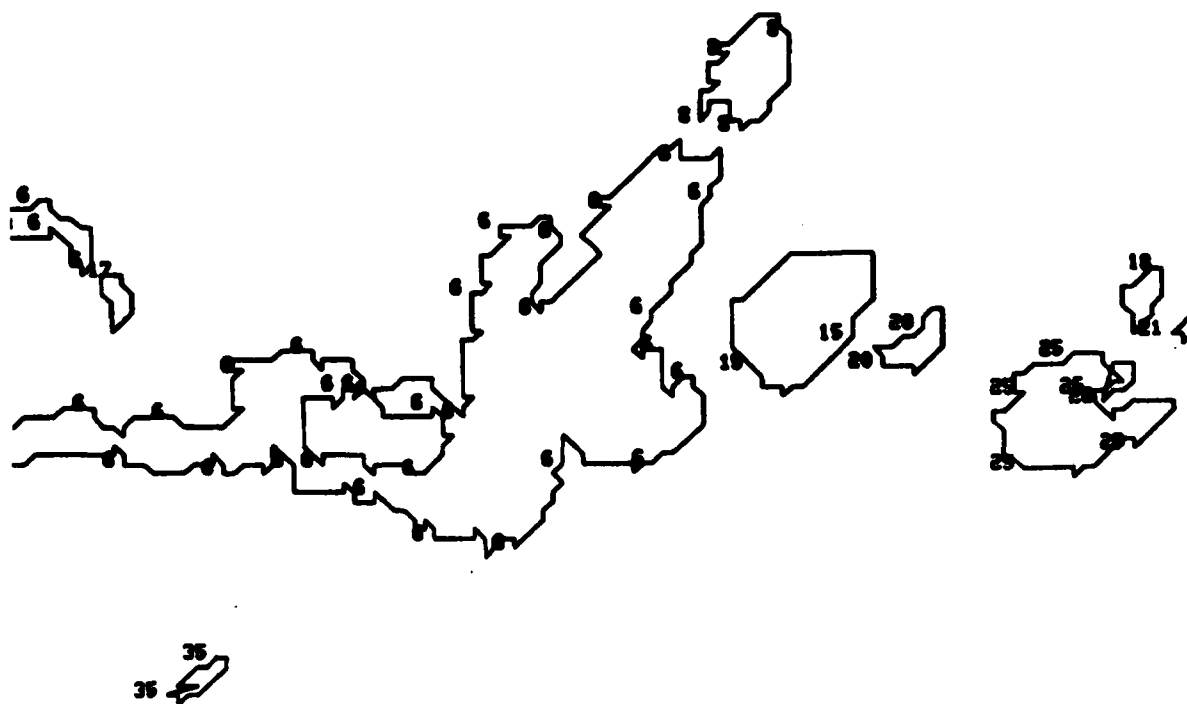


Figure 18. Polygon after closing by distance measure.

```

0024 70      CONTINUE ! JUMPS OUT OF LOOP HERE
0025      POINTV=I
0026      IF(POINTV.GT.(POINT)) GOTO 20 ! FINISHED
0027      POINTS=POINTS+1 ! UPDATE SEGMENT POINTER
0028      SEGS(POINTS,1)=I ! MARK BEGINNING OF SEGMENT
0029      SEGS(POINTS,2)=I
0030      COORDI(POINTV,5)=-1
0031 60      CONTINUE ! RETURN POINT FOR SEGMENT CONSTRUCTION
0032      L=POINTV ! INTERMEDIATE STORAGE LOCATION
0033      CALL WHERE(COORDI(L,1),COORDI(L,2),COORDI(L,3),COORDI(L,4),SEARCH)
C      SEARCH CONTAINS ACTIVE POINT FOR SEARCH
0034      CALL FIND(SEARCH,FINDP,POINTV,NP) ! FINDP POINTS TO NEXT OR 0
0035      IF(FINDP.EQ.0) GOTO 30 ! CANNOT CONTINUE
0036      IF(COORDI(FINDP,5).NE.0) GOTO 50
0037      COORDI(POINTV,5)=FINDP
0038      SEGS(POINTS,2)=FINDP
0039      COORDI(FINDP,5)=-1
0040      POINTV=FINDP
0041      GOTO 60
0042 50      CONTINUE! MERGE LISTS
0043      COORDI(POINTV,5)=SEGS(NP,1)
0044      SEGS(NP,1)=SEGS(POINTS,1)
0045      POINTS=POINTS-1
0046      GOTO 85 ! BEGIN A NEW SEGMENT
0047      COORDI(POINTV,5)=-1
0048      IF(SEGS(POINTS,1).EQ.SEGS(POINTS,2)) COORDI(POINTV,5)=0
0049      IF(SEGS(POINTS,1).EQ.SEGS(POINTS,2)) POINTS=POINTS-1
0050      GOTO 85
0051 20      CONTINUE
0052      K1=K1+1
0053      MIN=5000
0054      H=SEGS(K1,2)
0055      IF(H.EQ.-1) GOTO 110
0056      CALL WHERE(COORDI(H,1),COORDI(H,2),COORDI(H,3),COORDI(H,4),Y)
0057      DO 150 J=1,POINTS
0058      T=SEGS(J,1)
0059      IF(T.EQ.-1) GOTO 150
0060      DJ=IABS(COORDI(T,3)-Y(3))
0061      DI=IABS(COORDI(T,2)-Y(2))
0062      IDIS=DJ+DI
0063      IF(IDIS.LT.MIN) INDEX=J
0064      IF(IDIS.LT.MIN) MIN=IDIS
0065 150      CONTINUE
0066      IF(MIN.GT.3) GOTO 110
0067      ICONST=SEGS(INDEX,2)
0068      IF(SEGS(INDEX,1).EQ.COORDI(ICONST,5)) GOTO 110
0069      COORDI(SEGS(K1,2),5)=SEGS(INDEX,1)
0070      SEGS(K1,2)=SEGS(INDEX,2)
0071      IF(K1.EQ.INDEX) GOTO 110
0072      SEGS(INDEX,1)=-1
0073      SEGS(INDEX,2)=-1
0074      K1=K1-1
0075 110      CONTINUE

```



```

0076      IF(K1.NE.POINTS)GOTO20
0077  21    CONTINUE
0078      CALL INITT(160)
0079      CALL DWINDO(-50.,1000.,10.,850.)
0080      CALL CHR Siz(3)
0081      CALL COMPLE(LX)
0082      CALL DISPLA
0083      CALL CHR Siz(3)
0084      CALL FINITT(0,780)
          TYPE *,'$LX',LX
          C    ACCEPT*,IJO
          C    IF(IJO,EQ.0) STOP
0085  230    FORMAT(3X,6(I5,X))
0086      LTIME=1
0087  670    CONTINUE
0088      TYPE *,' * ? '
0089      READ(5,222) IJO
0090      IF(IJO,EQ.0) STOP
0091      IF(LTIME.EQ.1)CALL INTTT(160)
0092      LTIME=TIME+1
0093      CALL DWINDO(-50.,1000.,10.,850.)
0094      CALL CHR Siz(3)
0095      CALL CHR Siz(3)
0096      CALL DISPL1(IJO)
0097      CALL CHR Siz(3)
0098      CALL FINITT(0,780)
0099      IF(IJO.NE.0) GOTO 670
0100  222    FORMAT(I3)
0101      CALL CHR Siz(3)
          CCCCC DO 500 I=1,POINTS
          C      IF(SEGS(I,4).LT.20)GOTO 500
          C      IF(SEGS(I,1).EQ.-1)GOTO 500
          C      IHEAD=SEGS(I,1)
          C      L=SEGS(I,2)
          C      CALL WHERE(COORDI(L,1),COORDI(L,2),COORDI(L,3),COORDI(L,4),SEARCH)
          C      ICYCLE=TABS(COORDI(IHEAD,3)-SEARCH(3))

0001      SUBROUTINE FIND(ACTIVE,FINDP,POINTV,NP)
0002      INTEGER*2 ACTIVE(3) ! POINT FROM WHERE SEARCH IS MADE
0003      INTEGER*2 POINT
0004      INTEGER*2 TEST1,TEST2
0005      INTEGER*2 FINDP ! INDEX OF POINT FOUND OR ZERO
0006      INTEGER*2 COORDI(1500,6) ! DATA TO BE SEARCHED
0007      INTEGER*2 SEGS(500,4) ! SEGMENT POINTERS
0008      INTEGER*2 DI,DJ,DEL,POINTV
0009      INTEGER*2 NP ! INDEX OF SEGS FOR HEAD OF MERGE
0010      INTEGER*2 POINTS
0011      COMMON /DTA/COORDI
0012      COMMON /DISOC/SEGS,POINTS
0013      POINT=COORDI(1,6)
0014      FINDP=0
0015      JDIS=1
0016      JLIM=0
0017      J2=ACTIVE(2)

```

```

0018      I2=ACTIVE(3)
0019      DO 5 L=100,1,-1
0020      IF((POINTV+L).LT.1) GOTO 4
0021      IF(POINTV+L.GT.POINT) GOTO 4
0022      JN=COORDI(POINTV+L,2)
0023      IN=COORDI(POINTV+L,3)
0024      TEST1=IN-I2
0025      IF(IABS(TEST1).GT.JDIS) GOTO 4
0026      TEST2=JN-J2
0027      IF(IABS(TEST2).GT.JDIS) GOTO 4
0028      GOTO(10,20,30,40,50,60,70,80) ACTIVE(1)+1
0029 10    IF(TEST1.LT.JLIM) GOTO 4
0030      GOTO 90
0031 20    IF((TEST1.LT.JLIM).AND.(TEST2.GT.JLIM)) GOTO 4
0032      GOTO 90
0033 30    IF(TEST2.GT.JLIM) GOTO 4
0034      GOTO 90
0035 40    IF((TEST1.GT.JLIM).AND.(TEST2.GT.JLIM)) GOTO 4
0036      GOTO 90
0037 50    IF(TEST1.GT.JLIM) GOTO 4
0038      GOTO 90
0039 60    IF((TEST1.GT.JLIM).AND.(TEST2.LT.JLIM)) GOTO 4
0040      GOTO 90
0041 70    IF(TEST2.LT.JLIM) GOTO 4
0042      GOTO 90
0043 80    IF((TEST1.LT.JLIM).AND.(TEST2.LT.JLIM)) GOTO 4
0044      GOTO 90
0045 90    CONTINUE
0046      FINDP=POINTV+L
0047      IF(COORDI(POINTV+L+5).EQ.0) RETURN ! A SIMPLE CONSTRUCT
0048      DO 1 J=1,POINTS-1 ! IS CANDIDATE THE HEAD OF A LIST
0049      NP=J
0050      IF(SEGS(J,1).EQ.FINDP.AND.(COORDI(SEGS(J,2),5).EQ.-1)) RETURN ! IT IS T
0051 1      CONTINUE
0052 4      FINDP=0
0053      IF((POINTV-L).LT.1) GOTO 5
0054      IF((POINTV-L-1).GT.POINT) GOTO 5
0055      JN=COORDI(POINTV-L,2)
0056      IN=COORDI(POINTV-L,3)
0057      TEST=IN-I2
0058      IF(IABS(TEST1).GT.JDIS) GOTO 5
0059      TEST2=JN-J2
0060      IF(IABS(TEST2).GT.(JDIS) GOTO 5
0061      GOTO(100,200,300,400,500,600,700,800) ACTIVE(1)+1
0062 100    IF(TEST1.LT.JLIM) GOTO 5
0063      GOTO 900
0064 200    IF(TEST1.LT.JLIM).AND.(TEST2.GT.JLIM) GOTO 5
0065      GOTO 900
0066 300    IF(TEST2.GT.JLIM) GOTO 5
0067      GOTO 900
0068 400    IF((TEST1.GT.JLIM).AND.(TEST2.GT.JLIM)) GOTO 5
0069      GOTO 900
0070 500    IF(TEST1.GT.JLIM) GOTO 5

```

```

0071      GOTO 900
0072  600  IF((TEST1.GT.JLIM).AND.(TEST2.LT.JLIM)) GOTO 5
0073      GOTO 900
0074  700  IF(TEST2.LT.JLIM) GOTO 5
0075      GOTO 900
0076  800  IF((TEST1.LT.JLIM).AND.(TEST2.LT.JLIM)) GOTO 5
0077      GOTO 900
0078  900  CONTINUE
0079      FINDP=POINTV-L
0080      IF(COORDI(POINTV-L,5).EQ.0) RETURN ! A SIMPLE CONSTRUCT
0081      DO 2 J=1,POINTS-1
0082          NP=J
0083          IF(SEGS(J,1).EQ.FINDP.AND.(COORDI(SEGS(J,2),5).EQ.-1)) RETURN
0084  2      CONTINUE
0085          FINDP=0
0086  5      CONTINUE
0087          RETURN
0088      END

```

This process results, in most cases, in a closed curve that can be analyzed by the Fourier Classification process but there are also cases where the contours produced are not suitable for processing but must be first modified by an operator (Figure 4, 5) before they can be used. In this case, the next step in the processing allows an operator to interactively modify the contours produced so that they are closed curves and can be analyzed by the Fourier descriptor programs. Programs that are used to modify the computer-generated polygons are documented in Reference 2. Examples of how they work are in Figures 3, 4, 19 and 20.

FOURIER DESCRIPTOR METHODS

The library that is stored in the computer does not use the (x,y) coordinates of the polygons, but first transforms them via the Fast Fourier Transforms and stores the "Fourier Descriptors." These Fourier Descriptors are defined as follows: A closed curve can be thought of as a function of a complex variable, $z(t)$, parametrized by arc-length t . We can normalize and have the curve described by $z(t)$, $0 < t < 2\pi$. If we go around the contour more than once, we get a periodic function, which can be expanded in a convergent Fourier series. The Fourier Descriptor of the curve is defined to be the Complex Fourier series expansion of $z(t)$ which is given by the formula

$$z(t) = \sum_{n=-\infty}^{\infty} A(n) e^{int} \quad \text{where}$$

$$A(n) = \frac{1}{2\pi} \int_0^{2\pi} z(t) e^{-int} dt$$

(See Figure 21, 22)

Thus the Fourier Descriptors (32 of the $A(1)$) for each element in the library are computed and stored into memory. The contour of the unknown plane is then found, the Fourier coefficients for this unknown are calculated, and the angular data necessary is obtained by finding the element i in the library whose Fourier coefficients are closest to the unknowns.

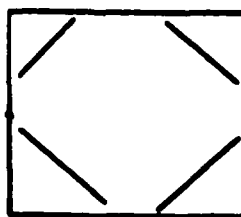
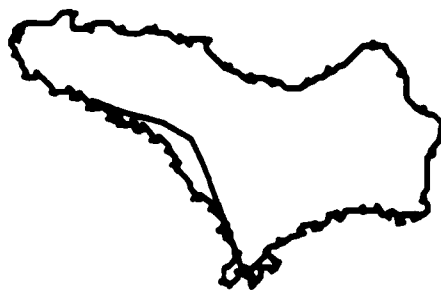
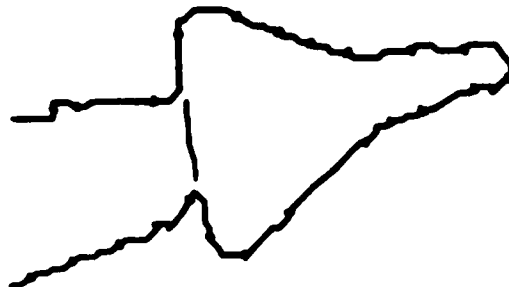


Figure 19. Polygons and corrections entered by the operator.

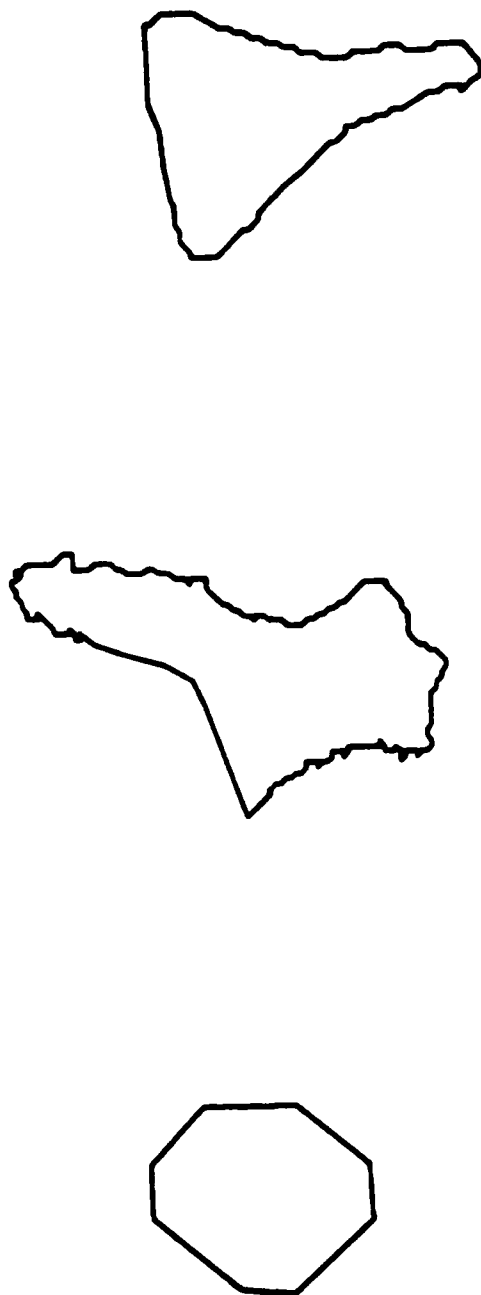
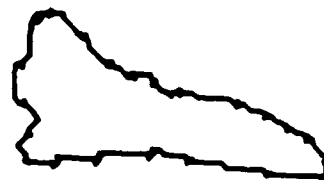
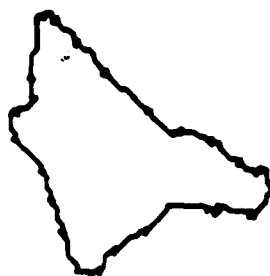


Figure 20. New polygons obtained from corrected polygons of Figure 19.

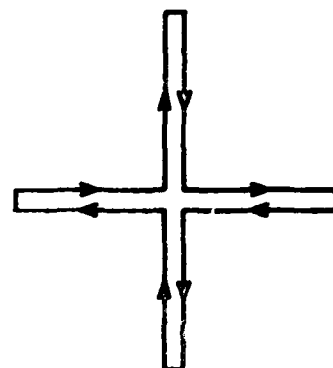
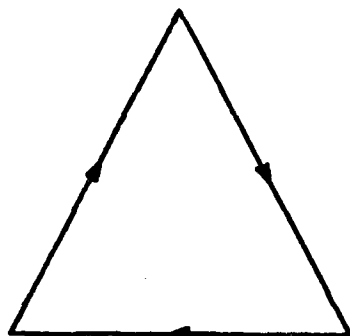
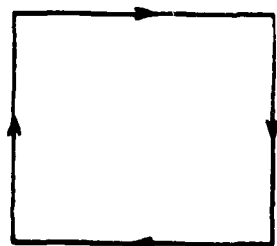


1.0000	6.2232
0.0780	3.2250
0.0000	0.0000
0.0046	0.0000
0.0122	4.1513
0.0000	0.0000
0.0031	6.1300
0.0000	5.0170
0.0103	0.0000
0.0000	6.0300
0.0007	0.0340
0.0001	0.0000
0.0040	5.0000
0.0021	0.0100
0.0000	6.1101
0.0011	0.0007
0.0012	0.4200
0.0013	3.6715
0.0032	1.0371
0.0019	2.7000
0.0000	2.5000
0.0040	3.0000
0.0033	1.0000
0.0007	0.1000
0.0000	6.0000
0.0007	2.0313
0.0007	5.0000
0.0172	5.0000
0.0000	2.0000
0.0004	0.0000
0.0003	6.0000

1.0000	0.0000
0.0581	1.1021
0.0371	1.5300
0.0760	0.0001
0.0100	3.4000
0.0173	0.0000
0.0004	0.1000
0.0003	1.5100
0.0100	0.0000
0.0030	4.1000
0.0100	2.0047
0.0000	5.0000
0.0030	0.0000
0.0000	0.0000
0.0030	0.0000
0.0007	5.0000
0.0015	2.0000
0.0000	5.0000
0.0031	3.0000
0.0100	0.1000
0.0040	5.0000
0.0000	0.0000
0.0000	2.1000
0.0000	5.0000
0.0114	5.0000
0.0000	5.0000
0.0001	5.0000
0.0000	0.0000
0.0001	2.0000
0.0001	0.0000
0.0013	5.0000

1.0000	0.0000
0.0123	4.5078
0.1362	0.0010
0.0356	0.5100
0.0126	5.0422
0.0000	0.5340
0.0001	5.6358
0.0044	0.4778
0.0100	0.1400
0.0000	3.0000
0.0004	0.6000
0.0000	4.0000
0.0000	1.0000
0.0003	5.0000
0.0040	2.0014
0.0000	0.2100
0.0020	2.0010
0.0070	1.1333
0.0021	5.5078
0.0043	2.1000
0.0000	5.1000
0.0130	1.0000
0.0000	5.1070
0.0072	2.1000
0.0150	0.0000
0.0103	1.0000
0.0070	0.4000
0.0370	6.2155
0.0000	2.1000
0.1700	5.0000
0.4000	6.0000

Figure 21. Three different contours of an F102 from video taken at WSMR. The Fourier coefficients are given in the format; absolute value, phase. The order is A(1), A(2), . . . , A(16), A(-15), . . . , A(-2), A(-1).



1.0000	0.0000
0.0008	6.2526
0.0052	4.9677
0.0008	6.2222
0.0379	6.2806
0.0008	6.1918
0.0023	5.2840
0.0008	6.1613
0.0103	6.2641
0.0008	6.1309
0.0015	5.5522
0.0008	6.1005
0.0040	6.2193
0.0008	6.0701
0.0012	5.7624
0.0008	6.0396
0.0027	0.1023
0.0008	0.2128
0.0013	0.6179
0.0008	0.1824
0.0063	0.0366
0.0008	0.1520
0.0018	0.8579
0.0008	0.1215
0.0183	0.0080
0.0008	0.0911
0.0032	1.1520
0.0008	0.0607
0.1091	0.0000
0.0008	0.0303
0.0155	1.4847

1.0000	6.2832
0.0131	3.2229
0.0144	0.0874
0.0594	6.2778
0.0068	3.2674
0.0070	0.1841
0.0172	6.2461
0.0041	3.2849
0.0044	0.2824
0.0070	6.1569
0.0027	3.2520
0.0030	0.3747
0.0032	5.9555
0.0019	3.1482
0.0022	0.4517
0.0020	5.5142
0.0017	3.2299
0.0024	0.4700
0.0029	5.9149
0.0024	3.0879
0.0051	0.1933
0.0042	5.9977
0.0037	3.0272
0.0121	0.0645
0.0067	6.0901
0.0062	3.0280
0.0362	0.0134
0.0131	6.1835
0.0136	3.0708
0.2461	6.2832
0.0785	6.2671

1.0000	6.2832
0.0019	1.8251
0.0139	0.0442
0.0320	6.2259
0.2060	6.2799
0.0231	3.0520
0.0069	2.0399
0.0043	5.7475
0.0192	0.0035
0.0023	2.1349
0.0012	5.8220
0.0074	6.0468
0.0184	0.0434
0.0069	2.8885
0.0024	3.4748
0.0010	4.3339
0.0031	3.6698
0.0011	5.7489
0.0019	2.1316
0.0067	3.0924
0.0154	6.1665
0.0071	6.2568
0.0044	0.5355
0.0021	6.1648
0.0022	4.0901
0.0009	1.0853
0.0090	2.9816
0.0315	3.1761
0.4984	6.2832
0.0624	0.0220
0.0461	0.0436

Figure 22. Three basic geometrical shapes and their Fourier coefficients.

In order to compute the Fourier Transforms, a closed curve description of the target to be analyzed has been produced by the computer and the operator. The description of this curve consists of a sequence of x,y coordinates, which are the vertices of a polygon. As the first step the length of this contour is computed, and the contour is resampled at a spacing chosen to make the total number of samples a power of two. This polygon is then filtered to remove noise, and the Fourier descriptor is computed by taking the Fast Fourier Transform of this sequence of (x,y) coordinates.

If two polygons are congruent in the plane then they can be shown to be so by a sequence of rotations, translations and contractions followed by a point-to-point comparison. If we have two congruent triangles represented by a sequence of x,y coordinates they can be shown to be congruent by first rotating both so that their longest side lies on the x axis, doing separate contractions so that they both have the same area then doing a point-by-point comparison starting at the greatest x coordinate. It is clear that the point-by-point comparison must be done starting at the same place on both triangles, and continuing at equidistantly sampled points in order for this process to be meaningful. The geometric transformations used to show two polygons congruent translate into the frequency domain as shown in Table I.

TABLE I. EQUIVALENT OPERATIONS

TIME DOMAIN	FREQUENCY DOMAIN
Translation	Addition to $a(0)$
Rotation	Multiplication of series by a constant
Comparison point change	Multiplication of $a(j)$ by $\exp(ijt)$

In order for a comparison to be meaningful in the frequency domain, a "normalization" in the frequency domain must be done similar to the geometric normalization that has been done for the triangles. This normalization¹ must be done, using only the operations which are listed on the right of Table I.

First, $a(0)$ is set equal to zero to normalize position. Size normalization is accomplished by dividing each coefficient by the absolute value of $a(1)$. To normalize the point where the comparison is to begin, we require that the phase of the two coefficients of largest magnitude be zero. For a simple closed curve that does not cut itself $a(1)$ is the coefficient of largest magnitude. Some polygons and their normalized Fourier coefficients appear in Figure 22.

Let $a(L)$ and $a(K)$ be two non-zero coefficients of the Fourier series. The normalization multiplicity of the coefficients $a(L)$ and $a(K)$ is defined to be $M = \text{abs}(K-L)$. Some of the geometric significance of M in the case where $a(1)$ and $a(L)$ are the only non-zero coefficients is given by the following proposition:

Let $z(t) = A(1) \cdot \exp(it) + A(L) \cdot \exp(iLt)$ with $\text{abs}(A(1)) > \text{abs}(A(L)) > 0$

1. T. P. Wallace and O. R. Mitchell, "Local and Global Shape Description of Two and Three Dimensional Objects," School of Electrical Engineering, Purdue University, September 1979.

PROPOSITION 1. If $\text{abs}(A(1)) > \text{abs}(L \cdot A(L))$ then the closed curve described by $z(t)$ has no intersections.

PROPOSITION 2. If $\text{abs}(A(1)) = \text{abs}(L \cdot A(L))$ then the closed curve described by $z(t)$ has $M = \text{abs}(1 - L)$ cusp points. The angles at these points are convex if $L < 0$ and concave if $L > 0$ (see Figure 23).

PROPOSITION 3. The function $\text{abs}(z(t))$ has M maximum points and M minimum points.

$$\text{Let } z(t) = \sum_{j=-\infty}^{j=\infty} A(j) \exp(ijt)$$

PROPOSITION 4. The requirement that $a(L)$ and $a(K)$ have zero-phase angle can be satisfied by M different orientation/starting point combinations.

PROPOSITION 5. $\max_j |A(j)|_{j=-\infty}^j = |A(1)|$ if the associated curve has no intersections.²

Thus for a figure whose second greatest coefficient is $a(L)$, there are $M = \text{abs}(L - 1)$ possible ways to normalize this figure. In order for an accurate comparison to be possible the normalization chosen for like figures must be the same. We use the following method to choose the normalization:

1. Calculate the Fourier coefficients for the M possible normalizations.
2. For each of the M normalizations calculate

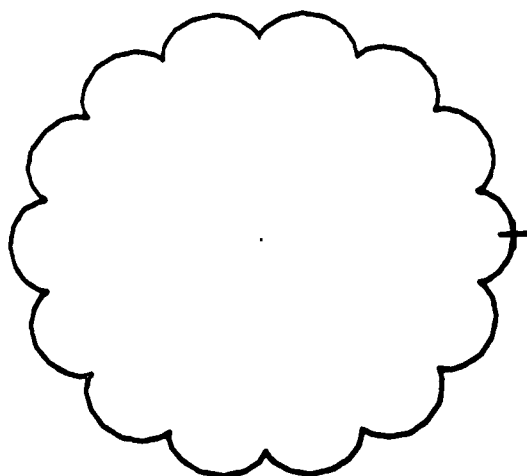
$$\sum \text{re}(a_i) |a_i|$$

3. Use the normalization which maximizes the above quantity.

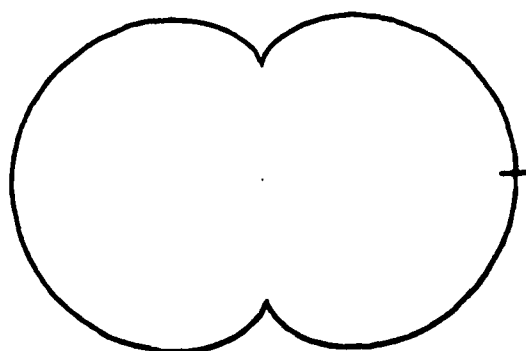
The pattern recognition method begins by constructing a three-dimensional representation of the target to be analyzed. A library of polygons is then constructed which are the projections of the three-dimensional object, as seen from different views. From this library of polygons a library of Fourier Descriptors is computed using normalization described above, and stored into the computer. When an unknown is to be analyzed and its contour is found, the Normalized Fourier Descriptors are calculated and these numbers are compared to the library entries via the difference.

$$\sum |a_i - \text{LIB}_j(i)|$$

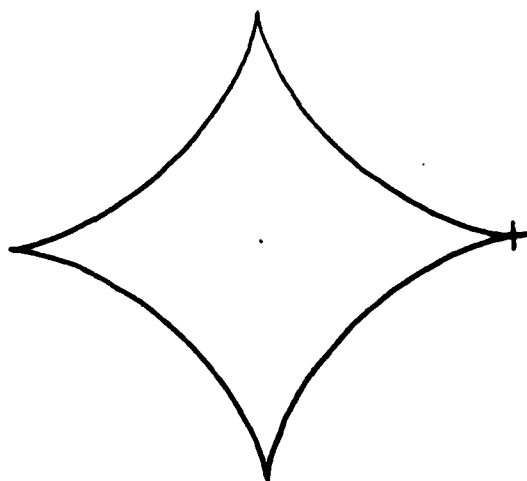
2. K. Phillips and R. Machuca, "The Geometry of Closed Curves Parametrized by Fourier Series," Research Memorandum, White Sands Missile Range, Instrumentation Directorate, Advanced Technology Office.



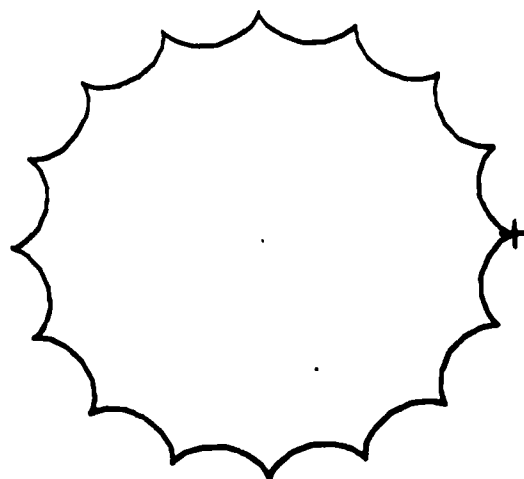
(a)



(b)



(c)



(d)

Figure 23. Contours generated by functions of the type $Z(t) = \exp(it) + 1/L \exp(iLt)$ for
 (a) $L = 15$ (b) $L = 3$ (c) $L = -3$ (d) $L = -15$

Once the closest element of the library is found, an interpolation³ is done to get an accurate measure of the aspect angles. When this is done the processing of the frame is finished.

We have described an interactive system that could be used to obtain aspect angle information from one frame of video. Before this system can be made completely automatic there must be research done with regards to two difficult problems. One is the automatic choosing of a threshold which would separate possible target points from the background. An approach being considered is an adaptive procedure for choosing τ where τ would be incremented if the size of the ellipse defined by the coefficients $a(1)$ and $a(-1)$ increased; and τ would be decreased if the area of this ellipse decreased. The other is the extraction of the target from the polygonal representation of the scene. Both are difficult problems which will require much research before a satisfactory solution can be found.

3. T. P. Wallace and O. R. Mitchel, ibid.

Appendix A

FINDING EDGES IN NOISY SCENES

Research into methods of identifying edges in a noisy scene has been an active field of investigation for many years. Treatment of the subject may be found in many books written over the past decade^{4,5,6} and many different approaches are proposed. Recently a survey and comparative analysis of the methods was made.⁷

The body of this appendix is segmented into four parts. In the first, we derive and define a "Moment Operator" which we show to work well for step and ramp edges. Then, we define and characterize second order edges using the concept of the rotation of a point in a vector field and develop the detector analytically. In Section 3 we develop the algorithms for implementing the previously defined operators. Finally, in Section 4, these algorithms are evaluated using ROC curves and compared with previously known techniques.

The detection of edges to isolate objects in a scene is motivated by many distinct problems. One such problem arises in a tracking system where the input video image is analyzed and the object to be tracked identified. Subsequent input and feedback to the drive controls causes the sensor to re-orient to a new position in an attempt to maintain the same x-y coordinate position for the object in the field of view. While this problem motivated the research that led to this paper, the results herein discussed are much broader in scope and application. The constraints imposed by this problem led to a method that is useful in high data throughput systems.

Section 1. Edges from Moments

First order edge detection methods work in the following way: A picture function $f(x,y)$ is transformed to another picture function $F(x,y) = Tf(x,y)$ in such a way that the edges of objects in the scene will be in the set $\{(x,y): F(x,y) > W\}$ for some W . The usual method is to transform the picture using T equal to the gradient operator. Different edge detection methods correspond to different numerical approximations to the gradient.

4. A. Rosenfeld and A. Kak, "Digital Picture Processing," Academic Press, New York, New York, 1976

5. B. Lipkin and A. Rosenfeld, "Picture Processing and Psychopictorics", Academic Press, New York, New York, 1970

6. W. Pratt, "Digital Image Processing," John Wiley and Sons, New York, New York, 1978

7. I. Abdou, "Quantitative Methods of Edge Detection," Image Processing Institute, University of Southern California, Los Angeles, California, 1978

The method used in our edge detection program is not based on derivatives. To reduce the effect of noise, this edge detection method uses integrals.

The reasoning for the use of moments to find edges is as follows:

A digitized picture can be thought as a lamina whose density at each point is $f(x,y)$, so points of high intensity correspond to points of high density.

A point (a,b) on an edge in the original function (see Figure A-1) would correspond to a point in this lamina (digitized picture) with high densities on one side and lower densities on the other side.

Thus if we look at a small lamina centered at point (a,b) and compute the center of mass of this small lamina, we can expect the center of mass to lie within an area of high densities.




 = regions of high density

Figure A-1. Example center of mass vectors for (1) and edge and (2) a region of uniform intensity.

Suppose we now look at a point (c,d) such that the densities around it are fairly constant. Then the center of mass of a small lamina about it would be close to (c,d) . In this case, a vector from (c,d) to the center of mass would be very small compared to a vector from (a,b) to the center of mass in the previous case.

We conclude that one way to transform $f(x,y)$ to $F(x,y)$ such that edges of the original picture lie in the set $F(x,y) > W$ is to replace every $f(x,y)$ by the length of the vector from (x,y) to the center of mass of a small lamina centered about (x,y) . That is, $F(x,y)$ is the magnitude of the vector from (x,y) to the center of gravity of a square lamina centered at (x,y) whose density is given by the picture function $f(x,y)$.

Figure A-2(b) is an example of how this method works on a scene (Figure A-2(a)) typical of those we study at WSMR.



(a) IMAGE OF ROCKET AND PLUME. THE PLUME IS THE LARGE REGION OF HIGHEST INTENSITY.



(b) RAMP AND STEP EDGES FOUND BY USING THE MOMENT OPERATOR.



(c) THE VECTOR FIELD GENERATED BY THE MOMENT OPERATOR.



(d) SECOND ORDER EDGES DETECTED BY USING THE VECTOR FIELD.

Figure A-2. Rocket and results of processing.

Once the coordinates (\bar{X}, \bar{Y}) of the center of mass of a lamina about (x, y) are calculated, the direction of the edge (if any) can easily be found. Since (\bar{X}, \bar{Y}) points to where the intensity of the picture is the highest, the direction of the edge is perpendicular to the direction of the vector from (x, y) to \bar{X}, \bar{Y} . If we take $(x, y) = (0, 0)$, then the direction of the edge is $\Theta = \text{Arctan} (\bar{Y}/\bar{X}) + \pi/2$.

Thus this model gives for each point in the scene a quantity that measures the probability that a point is an edge point and a direction which is the direction of a possible edge through that point.

The model introduced in Section I will not work for roof edges. This is because at the very peak of the roof, exactly where the edge is situated, both \bar{X} and \bar{Y} are equal to zero. In order to detect roof edges we need to take advantage of the direction information, and as Figures 6(a), (b) and (c) show we need to detect the shearing cause by the change in direction of the vector field at the edge points. One way of doing this is by using a tool from the theory of vector fields, namely the rotation of a vector field about a point.

If a curve Γ on the plane (scene) is given in the form

$$\Gamma: x = X(t), y = Y(t) \quad a \leq t \leq b$$

then $\Phi(t) = \{\phi[x(t), y(t)], \psi[x(t), y(t)]\}$ is defined on the interval $[a, b]$ (see Figure A-3).



Figure A-3. A curve Γ and its corresponding vector field $\Phi(t)$

For each $t \in [a, b]$ there is determined an angle, the angle in radians between $\Phi(t)$ and $\Phi(a)$ measured from $\Phi(a)$ to $\Phi(t)$. This angle is a many-valued function (vanishing for $t = a$) is designated by $\theta(t)$ and called an angular function of the field Φ on a curve Γ . The rotation of the field Φ on the curve Γ is defined to be

$$\gamma(\Phi, \Gamma) = \frac{1}{2\pi} [\theta(b) - \theta(a)]$$

If Γ is a closed Jordan curve, then the rotation is found by subdividing Γ into two curves (not closed), computing the rotation of each, and adding. In the following, Γ is taken to be a small circle about a point.

We can write the rotation as

$$\gamma = \frac{1}{2\pi} [\theta(b) - \theta(a)] = \frac{1}{2\pi} \int \frac{d\theta(t)}{dt} dt.$$

With $\theta(t) = \text{Arctan } Y/X + \pi/2$, we make the following observations:

If $\theta(t) = \text{constant}$, then $\frac{d\theta(t)}{dt} = 0$ and $\gamma = 0$. So $\gamma = 0$ when $x = \text{a point on the edge of an object in a scene (see Figure A-4).}$

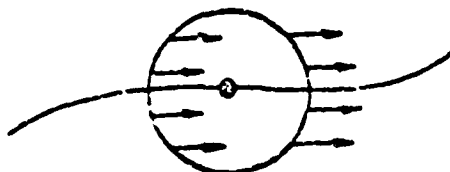


Figure A-4. Vector Field at a step or ramp edge point.

Section 2. Second Order Edges

After a scene is processed by the moment edge detector, each point is assigned a direction and a magnitude. In effect this specifies a vector at each point of the plan in question; i.e., these vectors define a vector field over the scene. An important tool in the study of vector fields is the rotation of a vector field.^{8,9} To define the rotation of a vector field, suppose a vector of the vector field ϕ at the point (x,y) is given by

$$\phi(x,y) = \{\phi(x,y), \psi(x,y)\}$$

$$\phi(x,y) = X(x,y)$$

$$\psi(x,y) = Y(x,y)$$

If θ is symmetric about x and Γ is a small circle about $x = \text{edge point}$ on a roof edge, see Figure 5, then write $\Gamma = \Gamma_1 + \Gamma_2$ (where $\Gamma_1 = \text{one half of the circle}$ and $\Gamma_2 = \text{the other half}$)

8. J. Milnor, "Topology from the differentiable viewpoint," University Press of Virginia, Charlottesville, Virginia, 1965

9. A. H. Stroud, "Approximate Calculation of Multiple Integrals," Prentice Hall, Englewood Hills, New Jersey, 1971

$$\int_{\Gamma} \frac{d\theta(t)}{dt} dt = \int_{\Gamma_1} d\theta(t) + \int_{\Gamma_2} d\theta(t) = \pi + \pi = 2\pi$$

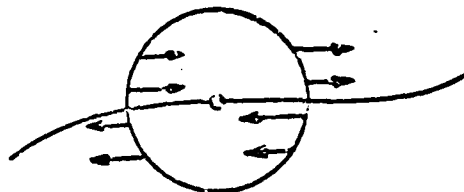


Figure A-5. Vector field at a roof edge point.

Figure A-6(a) and A-6(b) are examples of how these observations can be used to detect second order edges.

Section 3. Algorithms for Implementation

a. Calculation of Moment. Since we are interested in real time applications of these methods we simplify the calculation of \bar{X} and \bar{Y} by setting

$$M = \int_{-h}^h \int_{-k}^k f(x+t, y+u) dt du = 1$$

This can be justified by observing that $M/4hk$ is the average of the intensities over a small neighborhood of (x,y) and so this value can be approximated by the average value of intensities over the entire picture. This would then be just a scale factor and so could be left out.

To calculate the integrals involved we use an integral formula¹⁰ of order $O(h^6)$. The formula for integration is

$$\iint F(x,y) = \sum_{i=1}^9 W_i D_i \text{ with } W_{2k+1} = 25/324, W_{2k} = 10/81$$

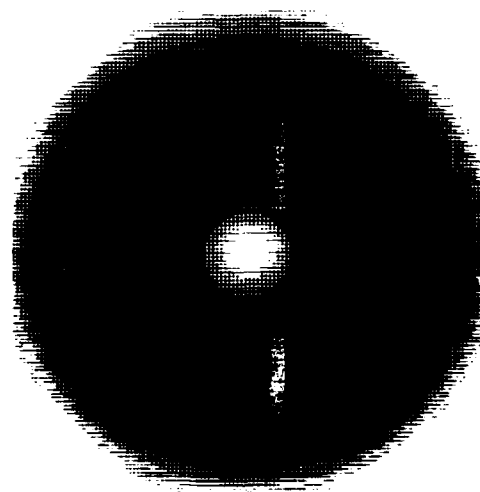
and if we apply this to the integrals for \bar{X} and \bar{Y} and factor out all scale factors we get

$$\bar{Y} = 5 * (D1 - D5) + 4 * (D8 + D2 - D6 - D4)$$

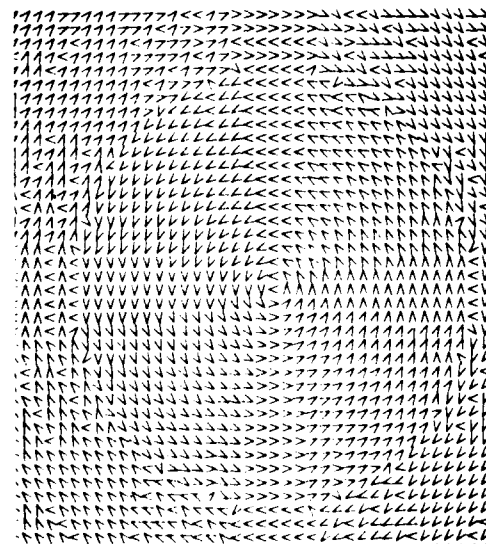
$$\bar{X} = 5 * (D7 - D3) + 4 * (D8 + D6 - D2 - D4)$$

and use $\text{abs}(X)^2 - \text{abs}(Y)^2$ for the associated magnitude. If we sweep a three-by-three window across digitized scene $D7$ can be taken as the upper

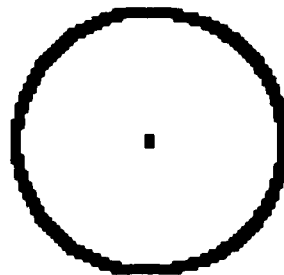
10. A. H. Stroud, ibid.



(a) ORIGINAL IMAGE OF DISK
WITH A ROOF EDGE.



(b) VECTOR FIELD GENERATED BY
APPLYING MOMENT OPERATOR
TO 2(a).



(c) EDGE POINTS OF 2(a) FOUND
BY IDENTIFYING THOSE
POINTS FOR WHICH $\int_C de = 2$.

Figure A-6. Original of roof edge and edge points.

left hand corner while D3 is the lower right hand corner. In this case the direction of a possible edge is equal to

$$\Theta = \text{Arctan} \left(\frac{Y - X}{Y + X} \right) + \pi/2$$

b. Calculation of the Rotation. The vector field of a roof edge will look like the vector field of Figure A-5. To find roof boundary points, we therefore have to find points for which, in a neighborhood of such a point,

$$\int_C d\Theta = 2\pi$$

The smallest region, in the discrete case over which we can take an integral, is a two-by-two window; thus our algorithm sweeps a two-by-two window across a scene and computes the integral $\int_C d\Theta$ for each of these windows. If it turns out that this integral is equal to 2π , those four points which make up the window are classified as boundary points. To calculate the integral of a two-by-two window we use an approximation

$$\int_C d\Theta \approx \sum_{\ell=1}^4 \Delta_{\ell} \Theta$$

computed by a computer program.¹¹

For the purpose of this experiment the procedure used to generate a file which is the file of detected second order edges in the following:

1. From the original file (scene) two files are generated; one (ACI) contains $\text{SQRT} [(X)^2 + (Y)^2]$; and the other (ANG) the angle of $(\Theta, 0 \leq \Theta \leq 255)$ a possible edge.

2. From the ANG and ACI files one new file AAA is created. AAA is created by sweeping a two-by-two window across the ANG file. The rotation is calculated, and if a point is classified as boundary, then to the corresponding point of AAA (initialized at zero) is added the average of those elements of ACI that have the same subscripts as those of the two-by-two window being swept across ANG.

Examples of how this method works are shown in Figure A-6.

11. R. Machuca and A. Gilbert, "Finding Edges in Noisy Scenes, IEEE Transactions on PAMI, unpublished.

Section 4. Evaluation

The methods described above were tested on disks whose edges were step, ramp and roof edges. The step and ramp edges had edge height equal to 16 while the roof edge was constructed by beginning at the center with gray value equal to 100 incrementing by one to gray value equal 132 and then decrementing by one to gray value 100. All files were 128 x 128 x 8.

To test the effectiveness of the different operations considered here we added Gaussian noise of different standard deviation to achieve a given signal to noise ratio and then tested the algorithms (Figure 7).

The SNR ratio was measured in db; that is, we used $SNR = 10 \log_{10} \left(\frac{16}{\sigma_n} \right)^2$

where σ_n = standard deviation of the noise. For the ramp and step edges we used¹² SNR = 4, 5, 6, . . . , 14 while for the roof edge the signal to noise ratios used were 10, 11, 12, . . . , 20. To measure the effectiveness of the different algorithms we graphed PF = the probability of false alarms vs. PD = the probability of detection (Figure A-9).¹² Figure A-8 contains examples of processed roof edge disks with SNR = 13. The graphs of PF vs. PD (ROC curves) for the corresponding operators appears in Figure A-8.

The results for different operators and step, ramp and roof edges appear respectively in Figure A-10. These graphs show that the performance of the moment operator is, in all cases, better than that of the Sobel operator. A significant improvement is obtained by first applying the average and then the moment operator. When the signal-to-noise ratio is high the median gives better results than the average; but there is a cross-over point at which the average filter gives better results than the median.

12. I. Abdou, "Quantitative Methods of Edge Detection," Image Processing Institute, University of Southern California, Los Angeles, California, 1978.

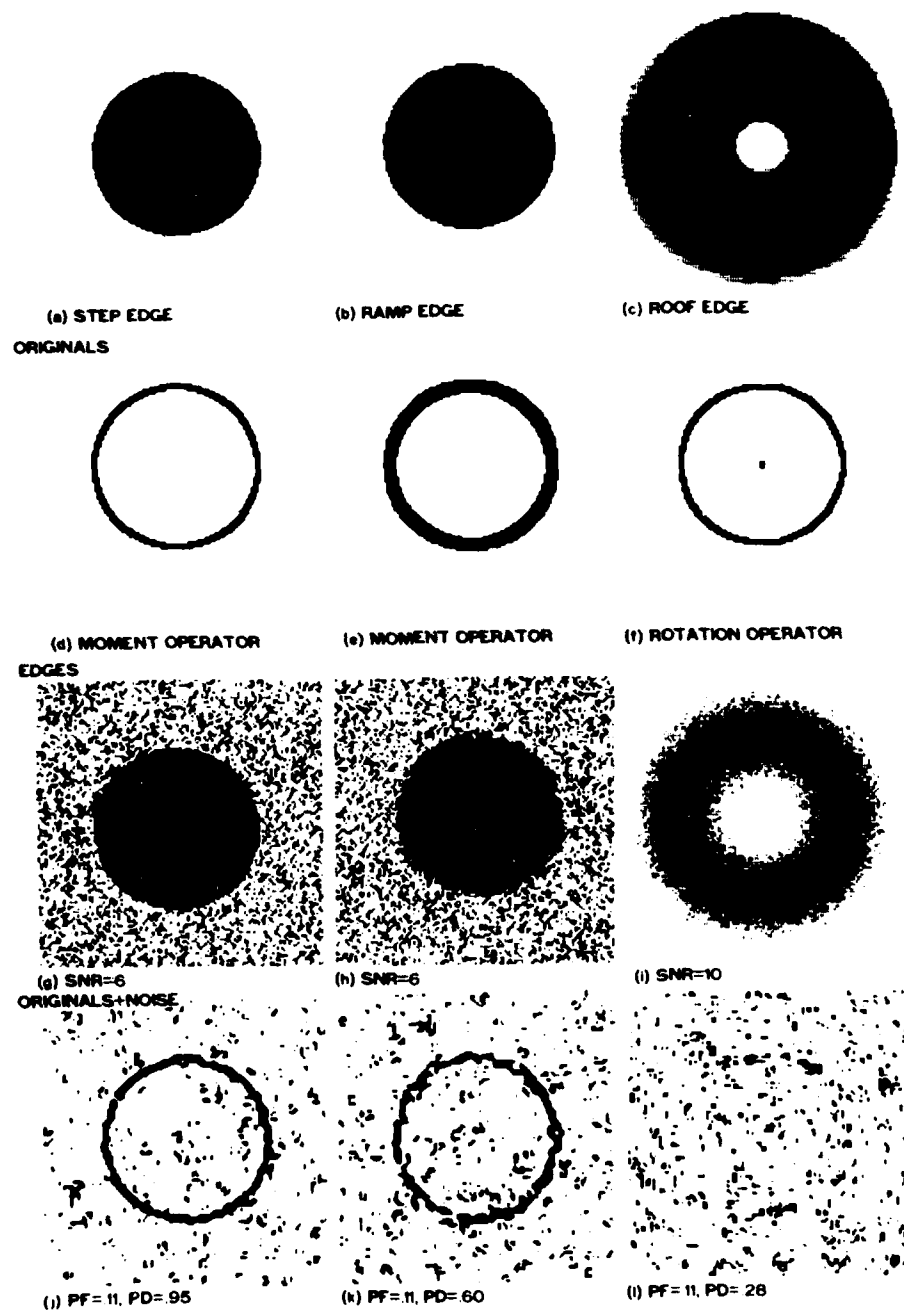


Figure 1-7. Clear edges and edges with noise added.

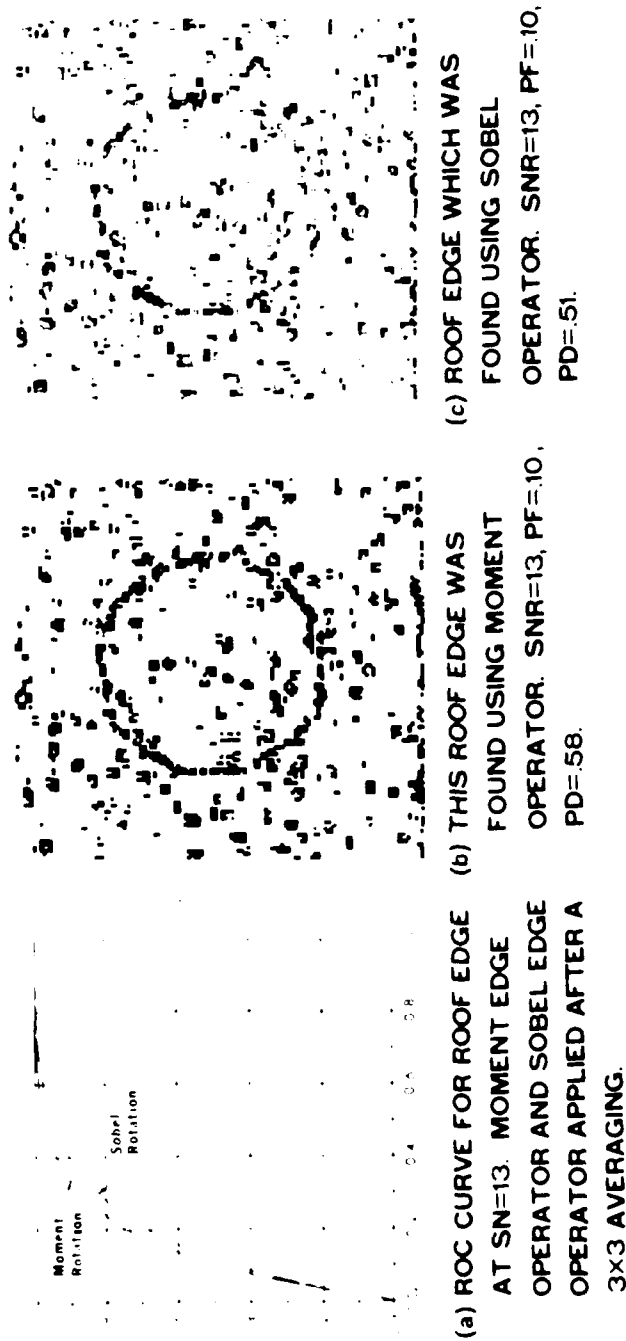


Figure A-8. Roof edges with corresponding ROC curves.

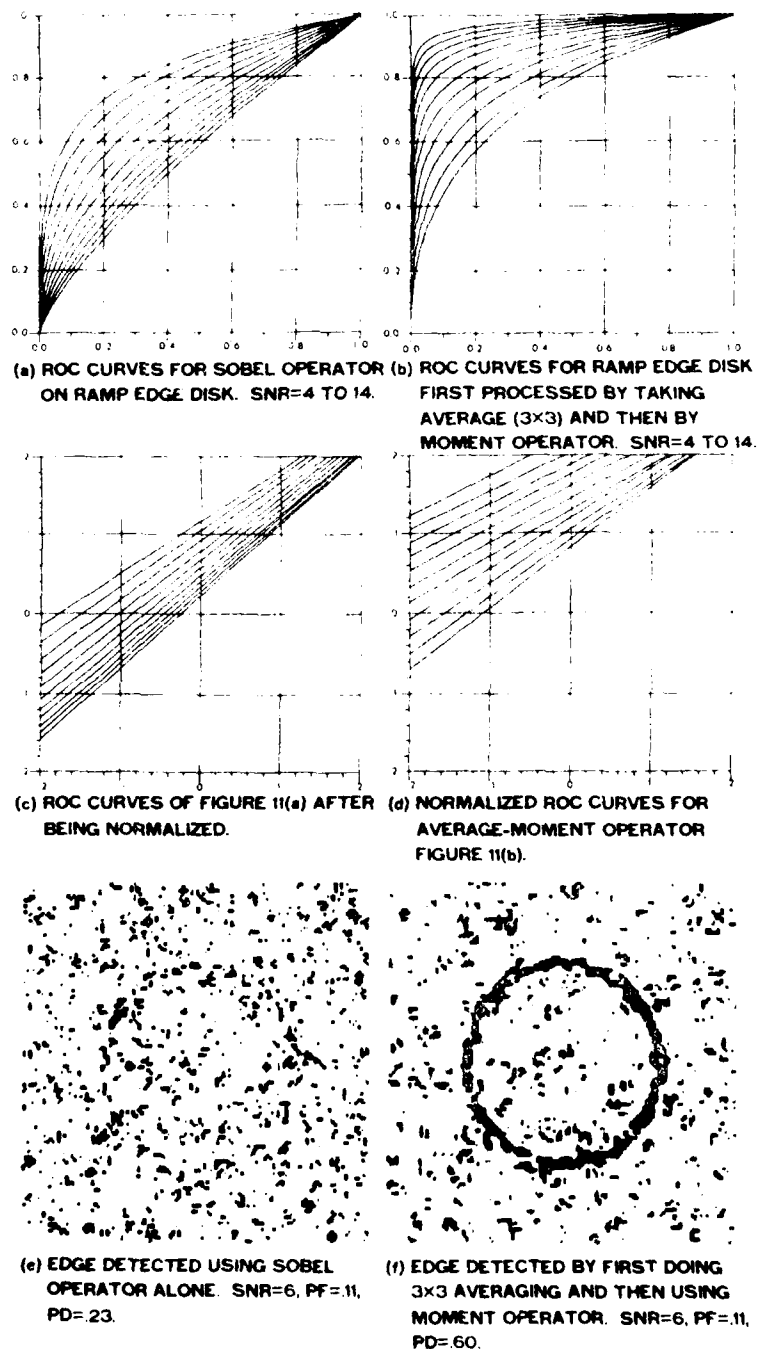


Figure A-9.. Comparison of Sobel and Moment operator.

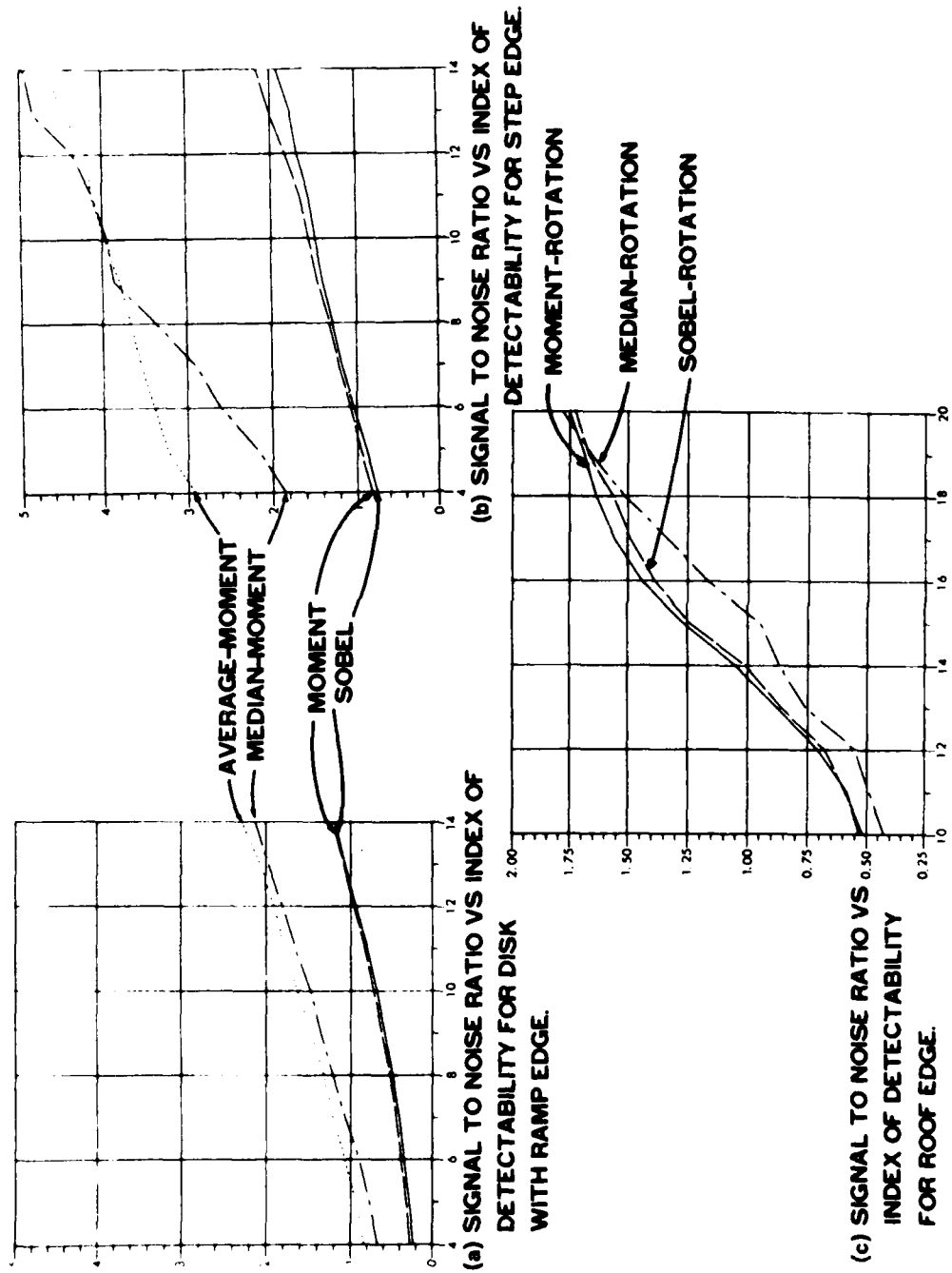


Figure A-10. Signal to noise ratio vs. index of detectability.

REFERENCES

1. T. P. Wallace and O. R. Mitchell, "Local and Global Shape Description of Two and Three Dimensional Objects," School of Electrical Engineering, Purdue University, September 1979.
2. K. Phillips and R. Machuca, "The Geometry of Closed Curves Parametized by Fourier Series," Research Memorandum, White Sands Missile Range, Instrumentation Directorate, Advanced Technology Office.
3. A. Rosenfeld and A. Kak, "Digital Picture Processing," Academic Press, New York, New York, 1976.
4. B. Lipkin and A. Rosenfeld, "Picture Processing and Psychopictorics," Academic Press, New York, New York, 1970.
5. W. Pratt, "Digital Image Processing," John Wiley and Sons, New York, New York, 1978.
6. I. Abdou, "Quantitative Methods of Edge Detection," Image Processing Institute, University of Southern California, Los Angeles, California, 1978.
7. J. Milnor, "Topology from the differentiable viewpoint," University Press of Virginia, Charlottesville, Virginia, 1965.
8. A. H. Stroud, "Approximate Calculation of Multiple Integrals," Prentice Hall, Englewood Hills, New Jersey, 1971.
9. R. Machuca and A. Gilbert, "Finding Edges in Noisy Scenes, IEEE Transactions on PAMI, unpublished.
10. I. Abdou, "Quantitative Methods of Edge Detection," Image Processing Institute, University of Southern California, Los Angeles, California, 1978.

DISTRIBUTION LIST

<u>Organization</u>	<u>Number of Copies</u>
STEWS-NR-A	1
CCNC-TWS	2
STEWS-NR-D	4
STEWS-PL	1
STEWS-PT-AL	3
STEWS-QA	1
STEWS-ID	1
STEWS-ID-D	1
STEWS-ID-O	1
STEWS-ID-E	1
STEWS-ID-P	3
STEWS-ID-T	1
STEWS-PT-AM	1
Commander US Army Test and Evaluation Command ATTN: DRSTE-AD-I Aberdeen Proving Ground, Maryland 21005	2
Commander Army Materiel Development and Readiness Command ATTN: DRCAD-P 5001 Eisenhower Avenue Alexandria, Virginia 22333	1
Director of Research and Development Headquarters, US Air Force Washington, DC 20315	1
Director US Naval Research Laboratory Department of the Navy ATTN: Code 463 Washington, DC 20390	1

DISTRIBUTION LIST (cont)

No. of Copies

Commander Air Force Cambridge Research Center L. G. Hanscom Field ATTN: AFCS Bedford, Massachusetts 01731	1
Commander US Naval Ordnance Test Station ATTN: Technical Library China Lake, California 93555	2
Director National Aeronautics and Space Administration ATTN: Technical Library Goddard Space Flight Center Greenbelt, Maryland 20771	2
AFATL/DLODL Eglin Air Force Base Florida 32542	1
Commander Pacific Missile Test Center Point Mugu, California 93041	1
Commanding Officer Naval Air Missile Test Center Point Mugu, California 93041	2
Office of the Chief Research and Development Department of the Army Washington, DC 20310	3
Commanding Officer US Army Electronics Command Meteorological Support Activity ATTN: Technical Library Fort Huachuca, Arizona 85613	2
Commanding Officer US Army Ballistics Research Laboratories Aberdeen Proving Ground, Maryland 21005	1
Commanding Officer US Army Research Office P. O. Box 12211 Research Triangle Park, North Carolina 27709	1

DISTRIBUTION LIST (cont)

No. of Copies

Commander Atlantic Missile Range Patrick Air Force Base, Florida 32925	1
Commanding Officer US Army Aviation Test Activity Edwards Air Force Base, California 93523	1
Defense Documentation Center Cameron Station Alexandria, Virginia 22314	12
US Army Materiel Systems Analysis Agency ATTN: DRXSYPMP Aberdeen Proving Ground, Maryland 21005	1
Director US Naval Research Laboratory Department of the Navy ATTN: Dr. J. R. Slagel Washington, DC 20378	1
Chief Department of the Air Force Mathematical Analysis Branch ATTN: Dr. H. L. Oestreicher Wright Patterson Air Force Base, Ohio 45433	1
Commander, Dept. of the Army US Army Research Office ATTN: Dr. Frank Kuhl, Bldg. 95N P.O. Box 12211 Dover, New Jersey 07801	1
Commander, US Army Research Office Department of the Army ATTN: Dr. W. Sander P. O. Box 12211 Research Triangle Park, North Carolina 27709	1
Commander, Dept. of the Army Office of the Assistant Secretary ATTN: Dr. J. H. Yang Washington, D.C. 20310	1

DISTRIBUTION LIST (Cont)

No. of copies

Commander, Dept. of the Army
Headquarters, US Army, TECOM
ATTN: Mr. Ben S. Goodwin
Aberdeen Proving Ground, Maryland
21005

1

Commander
Department of the Army
Office of the Assistant Secretary
ATTN: Dr. E Yore
Washington, D. C. 20310

1

Commander, Dept. of the Army
U. S. Army Armament Research and Development Command
ATTN: Dr. A. Gyoroc, EC & SCWSL
Dover, New Jersey 07801

1